WRDC-TR-90-8007
Volume VI
Part 2

INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume VI - Network Transaction Manager Subsystem
Part 2 - Network Transaction Manager (NTM) Programmer's Reference Manual

S. Barker

Control Data Corporation
Integration Technology Services
2970 Presidential Drive
Fairborn, OH 45324-6209

September 1990

Final Report for Period 1 April 1987 - 31 December 1990

MANUFACTURING TECHNOLOGY DIRECTORATE
WRIGHT RESEARCH AND DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, regardless whether or not the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data. It should not, therefore, be construed or implied by any person, persons, or organization that the Government is licensing or conveying any rights or permission to manufacture, use, or market any patented invention that may in any way be related thereto.

This technical report has been reviewed and is approved for publication.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations

DAVID L. JUDSON, Project Manager
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

FOR THE COMMANDER:

BRUCE A. RASMUSSEN, Chief
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

If your address has changed, if you wish to be removed form our mailing list, or if the addressee is no longer employed by your organization please notify WRDC/MTI, Wright-Patterson Air Force Base, OH 45433-6533 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release: Distribution is Unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) PRM620342000	5. MONITORING ORGANIZATION REPORT NUMBER(S) WRDC-TR-90-8007 Vol. VI, Part 2
6a. NAME OF PERFORMING ORGANIZATION Control Data Corporation; Integration Technology Services 6b. OFFICE SYMBOL *(if applicable)*	7a. NAME OF MONITORING ORGANIZATION WRDC/MTI
6c. ADDRESS *(City, State, and ZIP Code)* 2970 Presidential Drive Fairborn, OH 45324-6209	7b. ADDRESS *(City, State, and ZIP Code)* WPAFB, OH 45433-6533
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Wright Research and Development Center, Air Force Systems Command, USAF 8b. OFFICE SYMBOL *(if applicable)* WRDC/MTI	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUM. F33600-87-C-0464
8c. ADDRESS *(City, State, and ZIP Code)* Wright-Patterson AFB, Ohio 45433-6533	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
78011F	595600	F95600	20950607

11. TITLE *(Include Security Classification)*
See Block 19

12. PERSONAL AUTHOR(S)
Structural Dynamics Research Corporation: Barker, S.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Yr.,Mo.,Day)*	15. PAGE COUNT
Final Report	4/1/87-12/31/90	1990 September 30	141

16. SUPPLEMENTARY NOTATION
WRDC/MTI Project Priority 6203

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify block no.)*
FIELD	GROUP	SUB GR.	
1308	0905		

19. ABSTRACT *(Continue on reverse if necessary and identify block number)*

This Network Transaction Manager (NTM) Programmer's Rererence Manual describes the set of services (callable routines) provided in the NTM Application Program Interface library.

Block 11 (Cont) - INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
VOL VI - Network Transaction Manager Subsystem
Part 2 - Network Transaction Manager (NTM) Programmer's Reference Manual

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
UNCLASSIFIED/UNLIMITED x SAME AS RPT. DTIC USERS	Unclassified	
22a NAME OF RESPONSIBLE INDIVIDUAL David L. Judson	22b. TELEPHONE NO. *(Include Area Code)* (513) 255-7371	22c. OFFICE SYMBOL WRDC/MTI

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE

FOREWORD

This technical report covers work performed under Air Force Contract F33600-87-C-0464, DAPro Project. This contract is sponsored by the Manufacturing Technology Directorate, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Bruce A. Rasmussen, Branch Chief, Integration Technology Division, Manufacturing Technology Directorate, through Mr. David L. Judson, Project Manager. The Prime Contractor was Integration Technology Services, Software Programs Division, of the Control Data Corporation, Dayton, Ohio, under the direction of Mr. W. A. Osborne. The DAPro Project Manager for Control Data Corporation was Mr. Jimmy P. Maxwell.

The DAPro project was created to continue the development, test, and demonstration of the Integrated Information Support System (IISS). The IISS technology work comprises enhancements to IISS software and the establishment and operation of IISS test bed hardware and communications for developers and users.

The following list names the Control Data Corporation subcontractors and their contributing activities:

SUBCONTRACTOR	ROLE
Control Data Corporation	Responsible for the overall Common Data Model design development and implementation, IISS integration and test, and technology transfer of IISS.
D. Appleton Company	Responsible for providing software information services for the Common Data Model and IDEF1X integration methodology.
ONTEK	Responsible for defining and testing a representative integrated system base in Artificial Intelligence techniques to establish fitness for use.
Simpact Corporation	Responsible for Communication development.
Structural Dynamics Research Corporation	Responsible for User Interfaces, Virtual Terminal Interface, and Network Transaction Manager design, development, implementation, and support.
Arizona State University	Responsible for test bed operations and support.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SECTION 1

INTRODUCTION

The Network Transaction Manager (NTM) Programmer's Reference manual describes the set of services (callable routines) provided in the NTM Application Program Interface library.

This manual is intended for use by application programmers who wish to use the NTM Application Program Interface library routines to send and receive messages between application programs running within the NTM environment.

Section 2, the NTM Environment, provides a high level description of the architecture and components of the NTM.

Section 3, Application Process Interface, provides a high level description of the NTM services and gives guidelines for use of the services.

Section 4, NTM Services, describes each of the services that are available to NTM applications.

Section 5, NTM Version 1.1 Compatability Interface, describes services similar to version 1.1 that provide a calling interface for 1.1 applications to run under 2.0.

Section 6, Programming Guidelines for Queue Server Applications, provides information on support for specified NTM applications which are known as Queue Servers.

SECTION 2

THE NTM ENVIRONMENT

The purpose of this Section is to provide a high level overview of the Network Transaction Manager environment to the Application Programmer. Additional detail on the NTM environment can be found in the NTM System Programmer's Manual.

The Network Transaction Manager (NTM) provides a facility for process to process communication between application programs that reside on either the same or different computers The NTM Application Program Interface (API), which is described in this manual, provides the application programmer with a set of system independent callable routines (NTM services) that enable message passing, process activation and termination, and file transfer services within a distributed processing environment. By using the NTM services, the application programmer does not have to be concerned with the specifics of the operating system routines that provide interprocess communications services or the details of what communications protocol will be used to deliver the message. The NTM isolates the application program from the operating system by providing a standardized application interface layer. This layer also enhances the portability of applications that execute within the NTM environment by removing the otherwise system specific interprocess communication code that would be present if the NTM services were not available.

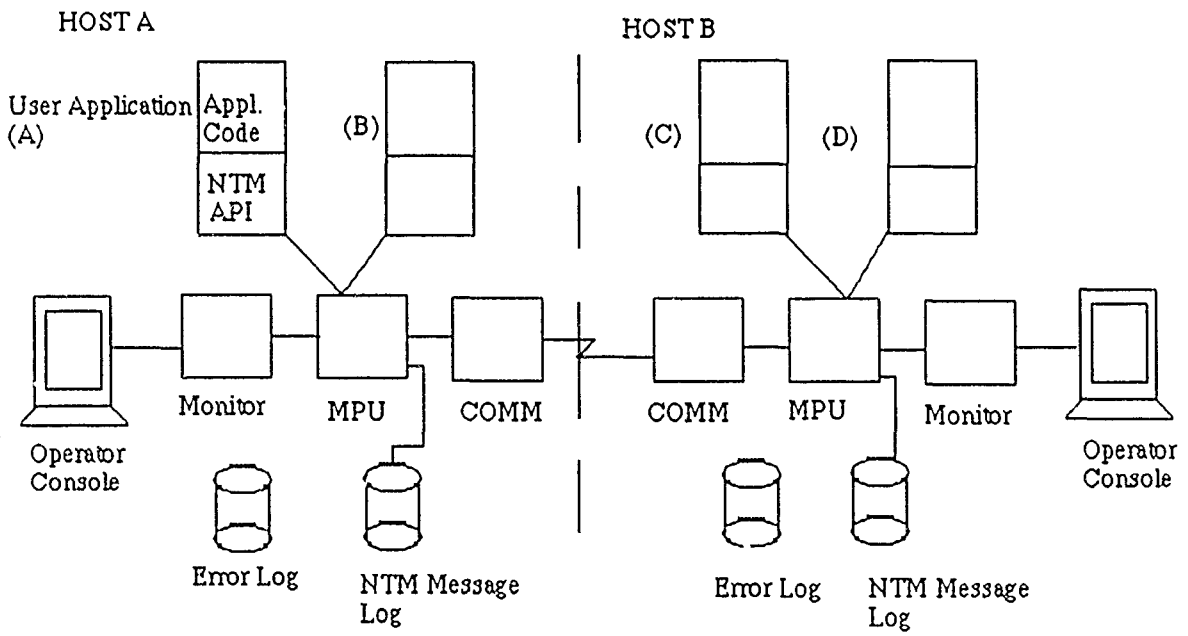


Figure 2-1 NTM Environment

This NTM environment, shown in Figure 2-1, consists of several different components which reside on each host and cooperate to perform the coordination, communication, and housekeeping functions necessary to enable application programs to send and receive messages. These components, which are in fact executing processes, are started when the NTM environment is initiated by the system operator (see the NTM Operator's Guide). The three main components of the NTM environment are.

- o Message Prccessing Unit (MPU)
- o Monitor
- o Communications Module (COMM)

To illustrate at a high level the role that each of these components play, it is helpful to trace the path that a message will take when a "send message" routine has been called by Application Program (A) that has specified a message to be delivered to Application Program (C), which in this case is executing on a different computer system then AP (A). When the SEND routine is invoked by Application Program (A), the specified message data is passed to the InterProcess Communication (IPC) of the Application Programmer's Interface that has been linked into the AP. The IPC routines then add a message header to the message data that identifies among other things the source and destination of the message. The message is then sent to a mailbox attached to the Message Processing Unit (MPU) process that is executing on the same host where AP (A) is running. The primary role of the MPU is to determine the destination of the message - whether it is an AP executing on the same host as the message sender, or the AP is located on a remote host. When the MPU determines that the message is to be sent off-host, the message is sent to a mailbox attached to the COMM module. The COMM module, which again is a separate process executing on the same machine as AP (A), is responsible for the management of link between the computer systems and the reliable delivery of NTM dependent communication routines, transforms the message into a data packet that is then placed on the physical link between the two computer systems. When the message is received by the COMM module executing on the remote machine, it is then delivered to the MPU which determines if the destination AP is currently executing. If in this case AP (C) was executing the message would be delivered to the mailbox that is attached to AP (C). Note that if AP (C) was not executing, the MPU would attempt to start it.

The Monitor Application Program is the NTM component which provides a central control point for an NTM environment. The Monitor AP is responsible for starting the other NTM components (MPU and COMM) upon NTM startup. Once these components are successfully started, the Monitor AP assumes the role of system monitor - handling system error messages, and processing commands received from the operators console. These commands enable the system operator to inquire on the status of the communication link or executing processes that are known to the NTM environment. Other commands enable the system operator to shut down the NTM environment or affect the environment in some way (see the NTM Operator's Guide for more detail).

In summary, the Network Transaction Manager Application Program Interface provides the application programmer with a set of high level callable routines that provide a system independent process to process communication facility that enables messages to be sent and received from other application programs that are known to the NTM without regard to the specifics of the computer platform on which the Application Program is running. The messages that are sent between APs can be used in a variety of ways - they can be used to signal that an event has occurred to another Application Program, they can request that a certain action be taken by a cooperating process, or they can be data that must be passed between Application Programs. The NTM components, MPU, Monitor, and COMM, provide a distributed process executive that enables the integration of application programs that may reside on heterogeneous computer systems.

SECTION 3

APPLICATION PROCESS INTERFACE

3.1 Overview of the NTM Services

The Application Program Interface (API) is a set of callable subroutines that are linked to each Application Program (AP) to provide the integration of the AP into the NTM. The API provides an application program with a set of services that enable it to send/receive messages to/from other application programs and NTM components. hese communicating application programs may reside on the same or different host computers.

The only restriction placed on an AP that is to be integrated into the NTM, and therefore use the message services of the NTM, is that the AP be written according to a format that includes NTM initiation (NTINIT) and termination (NTTERM) calls at the beginning and the end of its execution. The "NTINIT" and "NTTERM," respectively, provide the NTM connection and termination services. Invoking these routines communicates to the NTM that the application has either started or has terminated.

Figure 3-1 lists the NTM Service calls. There are four functional categories: Connection services, Communication services, Inquiry Services, and Privileged services. Most APs will use only the Connection and Communication services. The NTM Inquiry Services provide status information that can help an application program optimize its performance. They also offer processing services to application programs that have special NTM handling requirements. The privileged services are for applications requiring an external connection to the NTM.

CONNECTION SERVICES

Service Name	Function
NTINIT	Provides initiation services for an AP
NTTERM	Signals AP termination status

COMMUNICATION SERVICES

Service Name	Function
NTMCHK	Checks for any current messages (use RCV to retrieve)
NTISND	Sends an initiation request
NTNSND	Sends a message
NTQSND	Sends reply message (used by Queue Server APs only)
NTRECV	Receives message(s)
NTSERR	Notifies the NTM and Parent of an AP (non-fatal) error
NTSTMD	Switch NTM message test mode on or off

INQUIRY SERVICES

Service Name	Function
NTUGET	Get the user's name and Original Source AP name
NTSABT	Signal to NTM to abort an AP

PRIVILEGED SERVICES

Service Name	Function
NTUCHG	Change the users's role during a session
NTHSTS	Get the status of a specified host
NTXCON	Provide initiation services for External APs
NTULOF	Send NTM user Logoff information to NTM
NTULON	Send NTM user information to NTM
NTWHST	Request the name of the current host

Figure 3-1 NTM Services Calls

The basic Connection services ("NTINIT" and "NTTERM") and Communication services (sending messages and receiving messages) are described from a functional viewpoint in Sections 3.1.1 through 3.1.4. All of the services with their arguments and return codes are described in Section 4.

3.1.1 Initiation (NTINIT)

NTINIT sets up the NTM to provide message services to all APs. Messages are routed and delivered by the NTM through message queues. NTINIT provides the AP connection to the NTM and performs the initiation logic necessary to handle later NTM calls from the AP.

The NTM initiation service also provides system state information to APs that perform special logic on certain events (i.e., NTM startup, or for AP startup events such as "First Run of AP"). The system state message also provides the AP Interface with certain AP characteristics the AP supports.

3.1.2 SEND Messages ("NTNSND", "NTISND", "NTQSND")

APs may send messages to other APs by using the basic message delivery services of the NTM. Three different Send services are provided to APs. The first, "NTNSND", is used for normal AP-AP communication. The others, "NTISND" (specific initiation of a new AP instance), and "NTQSND" (queue-server reply message), all request special NTM message services.

The AP Interface returns the NTM's acceptance of each message to the AP which sent it. A successful return indicates that the NTM has accepted the message for delivery. A nonsuccessful return will indicate where the request failed NTM processing. Failure may be due to invalid calling arguments or NTM resource errors.

Some messages sent by an AP require responses from the receiving AP. These are called message pairs. The NTM provides message pairing support to the APs on the "NTNSND" and "NTISND". An AP can indicate that a particular message requires a response, and hence, pairing support from the NTM, by setting the timeout indicator argument of the "send message" services with an appropriate value. The NTM will provide the AP an indication when the elapsed time expires before a response has returned.

An AP can establish a test mode of operation with the "NTTSMD" service. The "TSTMOD" service allows the Test Mode indicator to be either enabled or disabled. The destination AP will receive the message with a "RCV" service whose return code will be set to indicate that the message was sent in test mode. The receiving AP then must handle the test-mode

message appropriately (i.e., perhaps inhibiting file updates). The NTTSMD service further enables the AP to receive messages signalling an AP error condition (NTSERR).

The "NTNSND" Service is used to send data that requires no special NTM handling beyond message pairing. The caller specifies the destination AP name and may use a logical channel specifier (see Section 3.1.5 and Section 5) to manage communications between the sending and receiving APs.

The "NTISND" Service is used when an AP is aware that its message will require the initiation of a new instance of the destination AP. It can also be used by APs who require concurrent access to multiple instances of the same AP. This service allows these APs to specifically request the initiation of a new instance of an AP. If the AP writer is in doubt whether to use an "NTISND" or "NTNSND", "NTNSND" should be the one chosen. The NTM can normally determine whether or not an AP requires an initiation, and will handle these messages accordingly.

The "NTISND" service must be used in the situation where the destination AP has the characteristic of requiring a specific initiation message. This restriction is placed on the AP by the Application Program developer when the AP is installed in the NTM. It serves to prevent the initiation of the AP upon receipt of unsolicited messages. This restriction applies only to initiation messages. Once the AP is running, it may receive any message from an authorized source AP using NTNSND service.

The "NTQSND" is the send service to be used by Queue-Server Applications to terminate connections with their parent APs. A Queue-server is a special type of application that requires information that is kept system (or at least machine) wide. Additional information about Queue Servers is available in Section 6 - Programming Guidelines for Queue Server Applications.

3.1.3 RECEIVE Messages

An AP receives messages from other APs in the NTM by using the "NTRECV" service. The AP has two basic options available to it when using the "NTRECV" service. These are a wait/no-wait option and a receive any message/receive specific message option. These options are described below.

WAIT/NO WAIT OPTION: If the AP uses a "NTRECV" with the wait argument set, its processing will be suspended until a message arrives (the message being either the message it was waiting for or an indicator of a time-out). On an "NTRECV" with "no wait", the AP will have control returned to it immediately. The return code on the "NTRECV" with "no wait" will either indicate that a message was received and can then be retrieved in the call's DATA argument or will indicate that no message of the type indicated has arrived for the AP.

RECEIVE ANY/RECEIVE SPECIFIC MESSAGE OPTION: An AP may set up to receive a message regardless of the source. the first message from a specific AP, the first from a specific AP on a specified channel (Section 3.1.5), or the first message from any AP on a specified channel. Also, the AP may use the message type parameter to increase the selectiveness of receiving messages. This feature provides flexibility to the AP programmer and off loads some of the AP's message bookkeeping and buffering. For example, if an AP has issued multiple "sends" and needs a response from a particular AP before it can continue, it can issue an "NTRECV" with a wait and the source specified. The AP NTM will buffer all of the AP's incoming messages until the requested response arrives and then return control and the requested message to the AP.

A companion service to "NTRECV" is "NTMCHK". It checks the AP's message to determine whether any messages have arrived for the AP. The AP can also check to see if a

message has arrived from a particular source channel by specifying one or both of these in the call. The AP can later retrieve any messages indicated by "NTMCHK" with the "NTRECV" service. The "NTMCHK" service should be particularly useful to AP programs that receive unsolicited messages. For example, an AP that can receive NTM shutdown messages could periodically use "NTMCHK" to determine if it has received a shutdown message.

3.1.4 Termination (NTTERM)

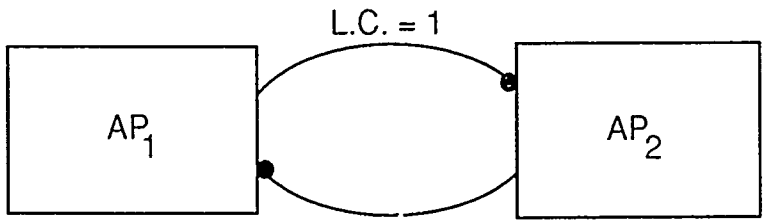
"NTTERM" disconnects the AP from the NTM, does any required cleanup and then terminates the AP. It should be the last executable statement in all application programs that use the NTM service routines. Once NTTERM is called, there is no return to the AP and execution ends during the NTTERM call.

3.1.5 AP-AP Communication - the Logical Channel Concept

The NTM provides a method of specifying the routing of a message called logical channel. Logical channel is an optional argument in the send and receive calls that can be used by the APs to:

- Pair Request and Response Messages. The source AP supplies a logical channel specifier with the destination AP's name on a send data request. The destination AP receives the logical channel specifier on the receive data call ("NTRECV") and uses it when it returns the response with its "NTNSND" service. (See Figure 3-2). In this way, a message pair can be confirmed by matching the pair source AP and logical channel. This is critical when the response comes from an AP that is not the destination specified in the original request message.

- Pair Multiple Outstanding Requests and Responses. The AP can use logical channel specifiers to manage multiple outstanding message pairs between one or several AP destinations (See Figures 3-3 and 3-4). The requesting AP manages the pairs by using the destination AP and logical channel to identify a unique pair. For example, in Figure 3-3, AP1 sends four messages to AP2, each on a different channel. AP2, whose protocol determines that it expects multiple messages from AP1, issues four consecutive "NTRECV" requests. On each of its NTRECV's, it obtains AP1's name and the logical channel specifier for the specific message. AP2 pairs the requests from AP1 to its responses by using the respective logical channel when it sends a response to AP1. This provides an easy and flexible pair management capability to the AP Programmer. Note that the AP must manage the assignment of the logical channel to the message.



AP1 specifies logical channel 1 on its "NTNSND" service

AP2 receives a message from AP1 on channel 1 and uses channel 1 when responding to AP1

Figure 3-2 Simple Paired Message Handling - Logical Channels

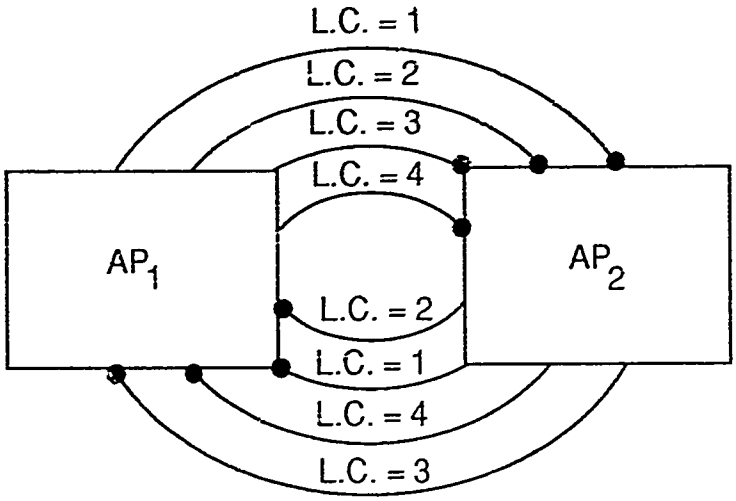


Figure 3-4 Multiple Pairs Between Two APs Using Logical Channels

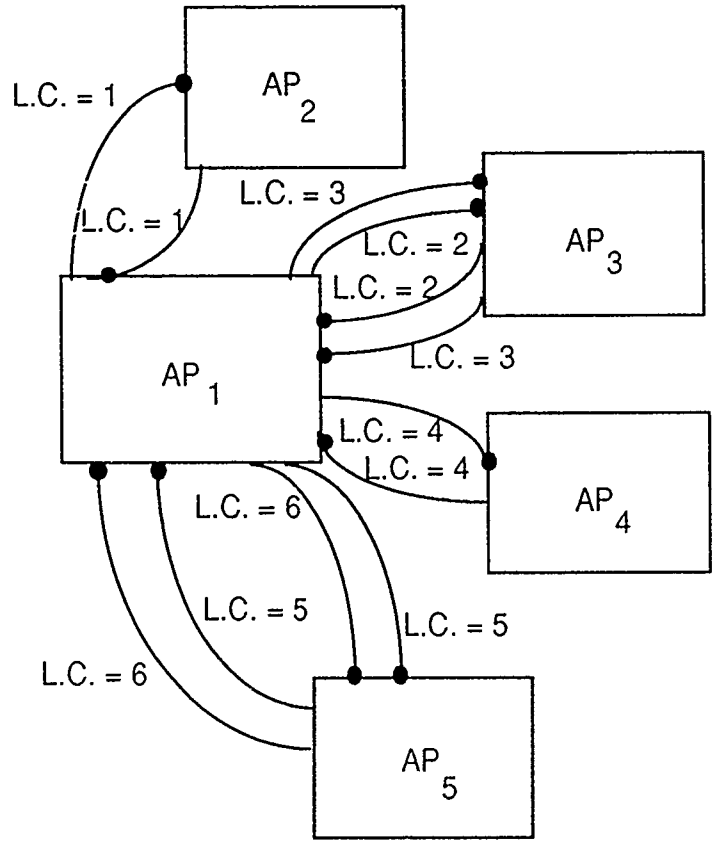


Figure 3-4 Multiple Pairs Between One AP and Many Destination APs

- Support AP Chaining. The AP can maintain a chain of communications between APs by using the Logical Channel specifier as the AP chain identifier (Figure 3-5). Each AP in the chain will use the same logical channel when sending messages to or receiving messages from any other AP in the chain. This function allows any AP in the chain to get the name of the chain's originating AP and the channel of this chain. With this information, any AP in the chain can send a message directly to the chain's originator.

- Maintain a Communication Path Between Two APs. An AP can maintain a "telephone like" conversation (Figure 3-6) with another AP by reserving one channel for this function. The AP supports this link by using the same logical channel on their "send" and "receive" calls. This is most useful to an AP that maintains simultaneous communication with multiple APs.

When a request message is sent, it is quite possible for the requesting AP to receive the response from an AP other than the destination specified in the original "send" call. As an example: AP1 requests data from AP2. AP2 does not have the data itself and in its turn spawns AP3 with instructions to obtain the data and send it directly to AP1. In this situation, the logical channel is the critical variable in matching the request and response message pair. Therefore, in order to support these message pairs it is necessary to place a restriction on the use of the channel numbers. This restriction requires that the logical channel of the response to the outstanding message match the logical channel of the outstanding message. This allows the NTM to ensure the integrity of its pairing and chaining logic. Figure 3-4 shows the supported use of unique channel numbers for pairs.

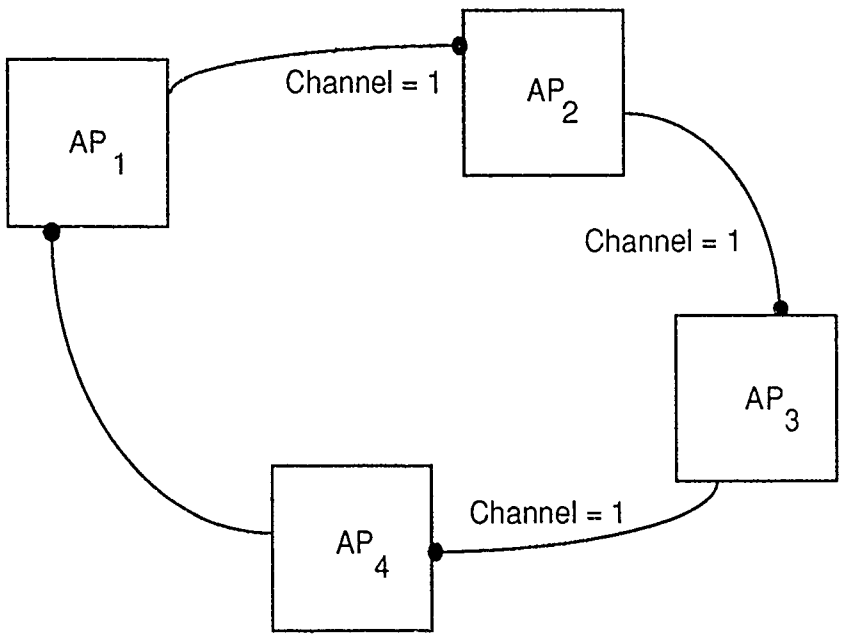


Figure 3-5 AP Chaining-Logical Channels

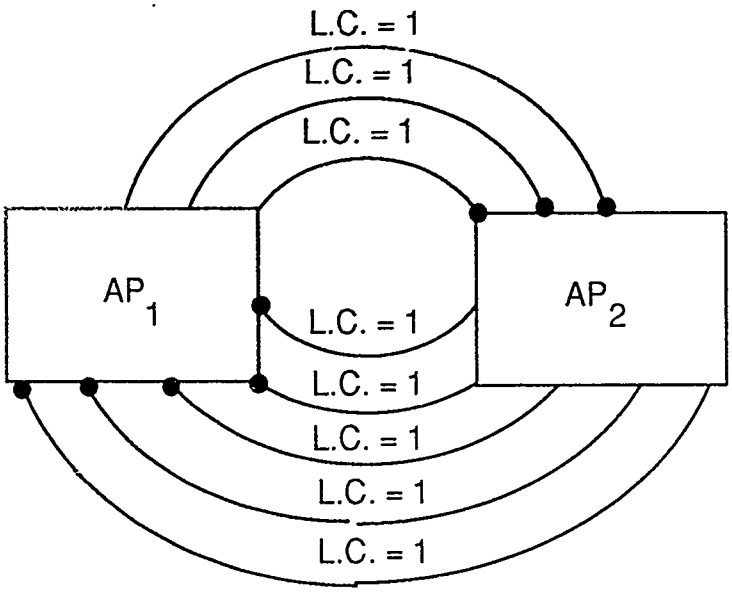


Figure 3-6 Maintain a Communication Path Using Logical/Channels

3.2 Writing New Application Processes Using the NTM

The following guidelines should be used when writing new applications using the NTM services.

1. The first executable statement in an application that will be started by the NTM must be a call to NTINIT. The last executable statement must be a call to NTTERM.

2. If the AP provides logic for special startup conditions (NTM startup), then it must include logic to test the startup status and perform the appropriate code indicated by the startup status return. (See NTINIT example, Section 4.)

3. If the AP expects to receive unsolicited messages (time-outs, shutdown requests, etc.), then it will periodically request these messages using the "NTRECV" service, or "NTMCHK" service followed by "NTRECV" service.

4. If the AP characteristic record (Appendix B) indicates that the AP does special shutdown processing on NTM shutdown, then the AP must also service unsolicited messages in the manner described above in Item 3. On recognition of a shutdown message, it must perform its shutdown logic and then transfer execution to the "NTTERM" service.

5. The AP must be linked or bound with the AP Interface routines for NTM execution (see Appendix D).

6. The AP must be integrated into the NTM by the NTM System Administrator. The AP writer must indicate the AP's characteristics (see Appendix B) and execution requirements to the NTM System Administrator.

7. The AP must supply the NTM the name of the APs to which it sends messages on the NTM "send" services. The names starting with "NT" are reserved.

8. All AP Interface service return codes are described in the file "NTMAPI.H". The legal values of the service's parameters are given in Section 4.4 of this document.

9. NTXCON is used by external AP's not started by the NTM. NTINIT is used by AP's that will be started by the NTM.

10. Applications calling NTHSTS or NTFCPY must be defined in APTTBL with two mailboxes.

SECTION 4

NTM SERVICES

4.1 Services Available to All NTM Applications

This section describes the following services which are available to all NTM applications.

NTHSTS
NTINIT
NTISND
NTMCHK
NTNSND
NTQSND
NTRECV
NTSABT
NTSERR
NTTERM
NTTSMD
NTUGET
NTWHST

The use of each service is explained in detail in the following sections. This includes a description of the service's purpose, return parameter values, the calling parameters' formats, and example calls. Note that Str and Int in the format mean character string and long integer, respectively.

A detailed description of the calling parameters with specific C definitions are contained in Section 4.3. The values of the service returns are defined in Section 4.4.

4.1.1 NTHSTS

Get the status of a specified HOST.

Calling Sequence

int nthsts (host_name)

Description

NTHSTS returns to the caller a status code containing relevant information about the HOST which was specified in the request.

Inputs

host_name	str (8)

Outputs

NONE

Service Return Code Values

NTM_HOST_UP
The specified host is active.

NTM_HOST_DOWN
The specified host is not active.

NTM_HOST_NOT_KNOWN
The specified host is not part of the NTM configuration.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

C LANGUAGE

```
static char host[9] = "IBM";
char rcode[5];

if hstats(host) != NTM_HOST_UP)
    printf("Host %s not available \n", host);
```

4.1.2 NTINIT

Provide initiation services for an AP.

Calling Sequence

int ntinit (system_state)

Description

NTINIT is the routine called by an AP to request that the AP interface perform the necessary initialization to allow the AP to execute and communicate with the NTM environment.

Inputs

NONE

Outputs

system_state	str (1)

Service Return Code Values

NTM_SUCCESS
The AP has successfully connected with the NTM.

NTM_FAILURE
The AP did not connect with the NTM.

NTM_RESOURCES_NOT_AVAILABLE
The connection was not made to the NTM. As this is a temporary condition, the call may be tried again.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

SYSTEM_STATE Values

NTM_INIT_NORMAL
The NTM is operating normally.

Examples

C LANGUAGE

```
char systate;

if (ntinit (&systate) != 0)
    {
    printf ("Unable to connect to NTM");
    ntterm(CONNECT_ERR);
    }
```

4.1.3 NTISND

Send a message that requests the specific initiation of a new instance of an AP. Data for the new destination AP instance may be included in the message.

Calling Sequence

```
int ntisnd (destination,logical_channel,timeout_value,binary_native_flag
        message_type,data_length,data)
```

Description

NTISND is used to specifically request the initiation of a new instance of a destination AP. Except for this feature, the services provided to an AP by this call are the same as NTNSND. This call is intended to support complex APs that must manage communications between multiple instances of the same AP. The NTISND user must specify different channel indicators to manage the communication between multiple instances of the same AP. The NTISND call may be used with or without data. (Also see Programming Guidelines for Queue-Server applications.)

The NTISND call must be used when starting APs having the characteristic of requiring a specific initiation. For APs having no restriction on initiation, the use of the NTISND call serves to guarantee the initiation of a new instance.

Inputs

destination	str(10)
logical_channel	str(3)
timeout_value	int
binary_native_flag	str (1)
message_type	str (2)
data_length	int
data	str (*)

LOGICAL_CHANNEL is optional. (Field should be blank if no channel is required.) TIMEOUT_VALUE should be set to zero if pairing support is not required. A non-zero timeout value will activate the message pairing processing. The Destination argument must include the directory prefix.

Outputs

NONE

Service Return Code Values

NTM_SUCCESS
The message sent under any of the SEND calls has been accepted by the NTM.

NTM_NOT_AUTHORIZED
The message failed the AUTHORIZED authorized check performed by the NTM. The source is not authorized to send a message of the given type to the destination.

NTM_INVALID_MESSAGE_TYPE
The message type argument is blank.

NTM_INVALID_DESTINATION
The destination AP name in the calling argument was not found in the NTM tables.

NTM_INVALID_DATA_LENGTH
The data length argument is blank.

NTM_INVALID_DATA_TYPE
The given binary-native flag is not valid. The value must be either "B" or "N".

NTM_RESOURCES_NOT_AVAILABLE
The resources needed by the NTM to process the message are not available.

NTM_INVALID_TIMEOUT_REQUEST
The given timeout value is not valid.

NTM_FAILURE
The NTM has rejected the message for reasons beyond the control of the source AP.

NTM_INVALID_SOURCE
The source AP name cannot be found in the NTM tables.

NTM_INVALID_LOGICAL_CHANNEL
The logical channel specified does not identify a valid queue server connection.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

C LANGUAGE

```
char destxfer[10];
char req_no[4];
static int timeout = 0;
int cmdlen;
char command[256];
char rcode[5];
int rc;

if ((rc = ntisnd(destxfer, req_no, timeout, "N", "MT", cmdlen, command)
    == NTM_SUCCESS)
    printf ("Message is on its way");
else
    printf ("ntisnd return code =%d", rc);
```

4.1.4 NTMCHK

Check for the arrival of messages in the AP's mailbox.

Calling Sequence

int ntmchk (source,logical_channel,message_type,data_length)

Description

NTMCHK can be used to determine whether either any message or a specific message has arrived at the AP. The size of the message that would be received is returned in the DATA_LENGTH parameter. The message is retrieved with the "NTRECV" service. NTMCHK does not deliver messages to the AP.

- o To check for any message, leave the logical_channel, message_type and source arguments blank. If more than one message is pending, the channel, type and source of the first message in the buffer will be returned.

- o To check for a message from a specific source on any channel, specify the source argument and leave the other arguments blank. If more than one message has arrived from the specified source, the channel and type of the first message in the buffer from the specified source will be returned.

- o To check for a message from a specific source on a specific channel, specify both source and logical channel. If any messages have arrived from this source on this channel, the return code will indicate NTM_SUCCESS.

- o To check for hot messages from the NTM (such as Shutdown Pending, Cancel Shutdown, or Shutdown), the message source must be specified as "NTM ". The seven following blanks are required by the NTM naming conventions.

Inputs or Outputs

source	str (10)
logical_channel	str (3)
message_type	str (2)

Outputs

data_length	int

Service Return Code Values

NTM_SUCCESS
The specified (or any, depending on values in the call parameters) message was found

NTM_NO_MESSAGE
The specified message was not found - or - if any, no message was found.

NTM_FAILURE
An error has occurred within the NTMCHK routine.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

C LANGUAGE

1. To check for any cold message:

```
memset (source, ' ', sizeof (source)):
memset (chan, ' ', sizeof (chan)):
if ntmchk (source, chan, length) == NTM_SUCCESS)
    rcv_msg();
```

2. To check for an NTM Message

```
memcpy (source, "NTM      ", sizeof (source));
memset (chan, ' ', sizeof (chan));
if ntmchk (source, chan, length) == NTM_SUCCESS
    rec_unsol_msg();
```

4.1.5 NTNSND

Send a message through the NTM.

Calling Sequence

```
int ntnsnd (destination,logical_channel,timeout_value,binary_native_flag,
        message_type,data_length,data)
```

Description

NTNSND is used to send a message that does not require special NTM handling from an AP to any authorized destination.

Inputs

destination	str (10)
logical_channel	str (3)
timeout_value	int
binary_native_flag	str (1)
message_type	str (2)
data_length	int
data	str (*)

LOGICAL_CHANNEL can be blank if a specific channel is not required. TIMEOUT_VALUE must be zero if pairing support is not required. If a non-zero timeout value is specified, the AP must specify a non-blank LOGICAL_CHANNEL in order to pair the responses or timeout message. The DESTINATION argument must specify the directory prefix. See also Programming Guidelines for Queue Server Applications.

Outputs

NONE

Service Return Code Values

NTM_SUCCESS
The message sent under any of the SEND calls has been accepted by the NTM.

NTM_NOT_AUTHORIZED
The message failed the AUTHORIZED authorized check performed by the NTM. The source is not authorized to send a message of the given type to the destination.

NTM_INVALID_MESSAGE_TYPE
The message type argument is blank.

NTM_INVALID_DESTINATION
The destination AP name in the calling argument was not found in the NTM tables.

NTM_INVALID_DATA_LENGTH
The data length argument is blank.

NTM_INVALID_DATA_TYPE
The given binary-native flag is not valid. The value must be either "B" or "N".

NTM_RESOURCES_NOT_AVAILABLE
The resources needed by the NTM to process the message are not available.

NTM_INVALID_TIMEOUT_REQUEST
The given timeout value is not valid.

NTM_FAILURE
The NTM has rejected the message for reasons beyond the control of the source AP.

NTM_INVALID_SOURCE
The source AP name cannot be found in the NTM tables.

NTM_INVALID_LOGICAL_CHANNEL
The logical channel specified does not identify a valid queue server connection.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

C LANGUAGE

```
char rmsgtyp[2];
char destination[10];
char log_chan[3];
static int zerotime = 0;
int lngth;
int rc;

if ((rc = ntnsnd(destination,log_chan,zerotime,"N",rmsgtyp,lngth,buff)
    == NTM_SUCCESS)
    printf ("Message not sent");
else
    printf ("NTISND return code =%d", rc);
```

4.1.6 NTQSND

Used by a queue server AP to send a message which terminates a connection between the queue server and its parent AP.

Calling Sequence

```
int ntqsnd (destination,logical_channel,timeout_value,binary_native_flag,
        message_type,data_length,data)
```

Description

NTQSND is the service used by a queue server AP to send the final message in a sequence of messages with an AP that has established a connection with the queue server (via NTISND). The "NTQSND" message terminates the connection between the queue server and its parent.

Inputs

destination	str (10)
logical_channel	str (3)
timeout_value	int
binary_native_flag	str (1)
message_type	str (2)
data_length	int
data	str (*)

Outputs

NONE

Service Return Code Values

NTM_SUCCESS
The message sent under any of the SEND calls has been accepted by the NTM.

NTM_NOT_AUTHORIZED
The message failed the AUTHORIZED authorized check performed by the NTM. The source is not authorized to send a message of the given type to the destination.

NTM_INVALID_MESSAGE_TYPE
The message type argument is blank.

NTM_INVALID_DESTINATION
The destination AP name in the calling argument was not found in the NTM tables.

NTM_INVALID_DATA_LENGTH
The data length argument is blank.

NTM_INVALID_DATA_TYPE
The given binary-native flag is not valid. The value must be either "B" or "N".

NTM_RESOURCES_NOT_AVAILABLE
The resources needed by the NTM to process the message are not available.

NTM_INVALID_TIMEOUT_REQUEST
The given timeout value is not valid.

NTM_FAILURE
The NTM has rejected the message for reasons beyond the control of the source AP.

NTM_INVALID_SOURCE
The source AP name cannot be found in the NTM tables.

NTM_INVALID_LOGICAL_CHANNEL
The logical channel specified does not identify a valid queue server connection.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

C LANGUAGE

```
char destxfer[12];
char req_no[4];
static int timeout = 0;
int cmdlen;
char command[256];
int rc;

if ((rc=ntqsnd(destxfer, req_no, timeout, "N", "MT", cmdlen, command))
    == NTM_SUCCESS)
    printf ("Message is on its way");
else
    printf ("QSEND return code = %d", rc);
```

4.1.7 NTRECV

Receive a message through the NTM.

Calling Sequence

int ntrecv (source,logical_channel,message_type,wait_flag,buffer_size,
data_length,data)

Description

RCV is used to receive a message, of any type, from any authorized source, including the NTM itself, via the services provided by the NTM. See also Programming Guidelines for Queue Server Applications.

Inputs

wait_flag	str (1)
buffer_size	int

Outputs

data_length	int
data	str (*)

Inputs/Outputs

message_type	str (2)
logical_channel	str (3)
source	str (10)

Service Return Code Values

NTM_SUCCESS
A message has been received.

NTM_REPLY_REQUIRED_MESSAGE
A paired message has been received.

NTM_NO_MESSAGE
No message was found in the AP's mailbox.

NTM_TIME_OUT_MESSAGE
A message time-out notification has arrived from the NTM.

NTM_TEST_MODE_MESSAGES
The message received is from an AP currently operating in test mode.

NTM_FAILURE
The NTM cannot access the Inter-Process Communication facilities.

NTM_BUFFER_TOO_SMALL
The buffer size specified is not large enough to hold the received message.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

NOTES

1. SOURCE, LOGICAL_CHANNEL, and/or MESSAGE_TYPE can be specified if selective message receiving is required.

2. If the AP wishes to specify the NTM for a source match, the source parameter must be entered as "NTM" on the call.

3. If the NTRECV service is called after a call to NTMCHK, the source and channel parameter should be those returned on the NTMCHK call.

4. If the AP expects to receive paired messages it must use the condition "NTM_REPLY_REQUIRED_MESSAGE" to test the NTRECV call return.

5. If "NTRECV" is called with no wait set, the AP must be able to deal with the possibility of a "NTM_NO_MESSAGE" return.

Examples

C LANGUAGE

```
char chan[4];
char destxfer[12];
char msgtyp[2];
static int bufsiz = 256;
int msglen;
char msgbuf[256];

if (ntrecv(destxfer, chan, msgtyp,  "1",bufsiz, msglen, msgbuf)
    != NTM_SUCCESS
    goto error;
```

4.1.8 NTSABT

Signal to the NTM that an AP is to be aborted.

Calling Sequence

int ntsabt (ap_name,logical_channel)

Description

NTSABT allows the calling AP to indicate to the NTM that the AP wishes to abort another AP which it (the caller) has spawned.

Inputs

ap_name	str (10)
logical_channel	str (3)

Outputs

NONE

Service Return Code Values

NTM_SUCCESS
The message to abort a child AP has been accepted for processing by the NTM.

NTM_FAILURE
The message to abort an AP has not been accepted by the NTM.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Notes

These code values only signify the acceptance of the abort message by the NTM. If the AP requires a specific acknowledgement of the actual abort, it must indicate this requirement when it defines its characteristics to the NTM system Administrator as described in the following paragraph.

An Application Program (AP) can request the NTM to abort another AP that the caller has spawned by using the NTM NTSABT call. The Return Code from the NTM will only indicate whether the NTSABT message has been accepted or rejected for processing by the NTM, not whether the Application Program has been aborted. The Application Program can determine if the AP to be aborted has terminated by waiting for NTM "Application End" messages after the call to NTSABT has been made. Note that these messages will be delivered to the requesting AP if the following conditions exist:

1. The requesting AP has been given characteristics in the Application Characteristics Table (NTMAPPS) of

On Child Abort	Send Message	"1"
On Child Termination	Send Message	"1"
On Child Shutdown	Send Message	"1"

2. The "to be" AP has been given the characteristics in the Application Characteristics Table (NTMAPPS) of

On Abort	Send Abort	"2"	or
On Abort	Abort	"3"	

Note that if the "to be aborted" AP has been given the On Abort Characteristic of Run to Completion (indicated by a "1" in NTMAPPS) then the AP will **not** be aborted and will not receive any notification of the NTSABT request.

Examples

C LANGUAGE

```
static char dest[10];
static char chan[4];

if ntsabt(dest, chan) != NTM_SUCCESS)
    printf ("Abort signal not accepted.")
else
    printf ("Abort signal accepted by the NTM.");
```

4.1.9 NTSERR

Allows an AP to signal the NTM and its original source when a non-fatal error occurs.

Calling Sequence

int ntserr (error_code,severity_level,error_description)

Description

NTSERR accepts error data from the calling AP and forwards that data to both the head of the calling AP's chain and the Monitor AP.

Inputs

error_code	str (5)
severity_level	str (1)
error_description	str (72)

Outputs

NONE

Service Return Code Values

NTM_SUCCESS
The NTSERR message has been sent successfully.

NTM_FAILURE
The NTSERR message could not be sent.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

C LANGUAGE

```
static char code[6] = "99925";
static char sev = 'W';
static char msg[61] = "Example of sigerr call";

if (ntserr(code, &sev, msg) != NTM_SUCCESS)
    printf ("NTSERR failed");
```

4.1.10 NTTERM

Signal AP termination status to the NTM.

Calling Sequence

int ntterm (termination_status)

Description

NTTERM performs termination processing for an executing AP and allows the AP to provide a status code to the NTM specifying the termination conditions.

Inputs

termination_status str (1)

Outputs

NONE

TERMINATION_STATUS Values

NTM_TERM_NORMAL
The AP has TERMINATION terminated normally.

NTM_TERM_SHUTDOWN
The AP has COMPLETE completed its shutdown processing.

NTM_TERM_ABORT
The AP has terminated as the result of a soft abort command.

NTM_TERM_EXCEPTION
The AP has terminated as the result of an internal (to the AP) exception condition.

NTM_TERM_INIT_FAIL
The AP has terminated due to an error incurred when trying to connect to the NTM.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

NOTES

1. The Termination Status values defined above must be used in this call. These values are used by the NTM to direct the clean-up procedures executed upon the AP's termination.

2. The call "NTTERM" with TERMINATION-STATUS set to NTM_TERM_INIT_FAIL must be used if the AP did not successfully connect to the NTM in call "NTINIT."

3. Any value other than those above will be treated as "1".

Examples

C LANGUAGE

```
ntterm (PROGRAM_ERR);
```

4.1.11 NTTSMD

Switch NTM message test mode on or off. When test mode is on, the calling AP will be able to receive asynchronous error messages. When test mode is off, these error messages will be discarded.

Calling Sequence

int nttsmd (test_status)

Description

NTTSMD is used by a calling AP to indicate to the NTM whether or not it wishes to receive asynchronous error messages.

Inputs

test_status str (1)

Values:

"0" = Test-Mode off
"1" = Test-Mode on
"2" = Fatal Error Messages only

Outputs

NONE

Service Return Code Values

NTM_TEST_MODE_ON
The test mode value for subsequent messages has been set to "on."

NTM_TEST_MODE_-OFF
The test mode value for subsequent messages has been set to "off."

NTM_TEST_MODE_FATAL
The test mode value for subsequent messages has been set for fatal messages only.

NTM_FAILURE
The test status value given in the calling argument is not recognized by the service.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

C LANGUAGE

```
char dbcode;

dif (nttsmd(&dbcode) != NTM_FAILURE)
    printf ("Test mode established");
```

4.1.12 NTUGET

Determine the AP Name and user name of the original node or source in an AP chain. The User Logon information is determined only when the original source AP has performed a call to NTULON.

Calling Sequence

int ntuget (ap_name,logical_channel,user_name,role_name,terminal_id)

Description

The parameters that NTUGET returns to the caller are the current user name, AP name and logical channel associated with its original source. This call allows any AP in a chain to determine its originating source and chain communication channel. If the originating source is a user at a terminal, a user-name and the name of the associated AP are returned.

If the originating source is an AP (not a terminal user), then the user Logon data will be blank on return. Only the AP_Name and Logical_Channel will have non-blank values.

Inputs

None

Outputs

ap_name	str (10)
logical_channel	str (3)
user_name	str (8)
role_name	str (10)
terminal_id	str (8)

Service Return Code Values

NTM_SUCCESS
The NTUGET service has successfully obtained all of the requested data.

NTM_FAILURE
The NTUGET service was not able to obtain the requested data.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

C LANGUAGE

```
if ntuget (apname, chan, user, role, term) == NTM_SUCCESS
    {
    strcpy (data, "xtsap5 has done its job");
    if (ntnsnd (apname, chan, 0, "N", "DM", length, data) != NTM_SUCCESS)
        {
        goof-in-program ("4");
        }
    }
else
    {
    goof in program ("8");
    }
```

4.1.13 NTWHST

Request the name of the Host on which a given AP resides.

Calling Sequence

int ntwhst (ap_name,host_name)

Description

NTWHST will provide the caller with the name of the Host machine that the specified AP currently resides on.

Inputs

ap_name str (10)

(If blank, the NTM will return the name of the host on which the calling AP resides)

Outputs

host_name str (8)

Service Return Code Values

NTM_SUCCESS
The host name for the given AP has been found.

NTM_AP_NOT_FOUND
The given AP name was not found in the NTM's tables.

NTM_FAILURE
The message requesting the AP's Host Name could not be sent.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

C LANGUAGE

```
static char ap_name[11] = "TSTAPPL";
static char host_name[9];
   .
if (ntwhst(ap_name, host_name) != NTM_SUCCESS)
    printf ("Application not fount in NTM tables");
```

4.2 Service Calls Available to Specialized NTM Applications

These are applications requesting NTM services but not started by the NTM, such as device drivers.

4.2.1 NTUCHG

Changes the user's role name during a logon session. This service assumes that an authorization check on the new role has been performed prior to making the change.

Calling Sequence

int ntuchg (terminal_id,user_name,new_role_name)

Description

The new role name is given to the NTM where it replaces the previous role name in the logon table.

Inputs

terminal_id	str (8)
user_name	str (8)
new_role_name	str (10)

Outputs

NONE

Service Return Code Values

NTM_SUCCESS
The message informing the Monitor AP of the new role name has been sent successfully.

NTM_FAILURE
The message to the Monitor AP was not sent. The new role has not been entered in the Logon Table.

NTM_USER_NOT_LOGGED_ON
The user is not logged on to the NTM. The new role has not been entered in the Logon Table.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

. Examples

C LANGUAGE

```
static char termid[9] = "TERMINAL";
static char usrnam[9] = "USERNAME";
static char rolenam[11] = "USERROLE01";
char rcode[5];

if (ntuchg(termid,usrnam,rolenam) != NTM_SUCCESS)
    printf ("Role change failed");
else
    printf ("Role change successful");
```

4.2.2 NTULOF

Allows an AP to inform the NTM of a user logoff.

Calling Sequence

int ntulof (terminal_id,user_name,role_name)

Description

NTULOF is used by the application to inform the NTM of a user logoff.

Inputs

terminal_id	str (8)
user_name	str (8)
role_name	str (10)

Outputs

NONE

Service Return Code Values

NTM_SUCCESS
The Monitor AP has been sent a message informing it that the User has logged off.

NTM_FAILURE
The message to the Monitor AP was not sent.

NTM_USER_NOT_LOGGED_ON
The user is not logged on to the NTM. No message is sent.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

C LANGUAGE

```
static char termid[9] = "TERMINAL";
static char usrnam[9] = "USERNAME";
static char rolenam[11] = "USERROLE01";

if (ntulof(termid,usrnam,rolenam) != NTM_SUCCESS)
    printf ("Logoff failed");
else
    printf ("Logoff successful");
```

4.2.3 NTULON

Allows an AP to pass logon information to the NTM.

Calling Sequence

int ntulon (terminal_id,user_name,role_name)

Description

LOGON is used by an AP to provide the NTM with the logon "user" information which the NTM needs to build its logon table. The NTM assumes that the AP has already performed the required authority checks on the user.

Inputs

terminal_id	str (8)
user_name	str (8)
role_name	str (10)

Outputs

NONE

Service Return Code Values

NTM_SUCCESS
The Monitor AP has successfully entered the Logon data in the Logon Table.

NTM_FAILURE
The Monitor AP could not write the new entry to the Logon Table.

NTM_USER_ALREADY_LOGGED_ON
The user is already logged on to the NTM.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

C LANGUAGE

```
static char termid[9] = "TERMINAL";
static char usrnam[9] = "USERNAME";
static char rolenam[11] = "USERROLE01";

if (ntulon(termid,usrnam,rolenam) != NTM_SUCCESS)
    printf ("Logon failed");
else
    printf ("Logon successful");
```

4.2.4 NTXCON

Provide initiation services for an AP.

Calling Sequence

int ntxcon()

Description

NTXCON is a routine called by APs that requires special NTM connection service. It requests that the AP interface perform the necessary initialization to allow the special AP to connect to and communicate with the NTM.

Inputs

NONE

Outputs

NONE

Service Return Code Values

NTM_SUCCESS
The AP has successfully connected to the NTM.

NTM_FAILURE
The AP could not connect to the NTM.

NTM_RESOURCES_NOT_AVAILABLE
The connection was not made to the NTM. As this is a temporary condition, NTXCON may be called again.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

C LANGUAGE

```
if (ntxcon) != NTM_SUCCESS)
    {
    printf("Unable to connect to NTM");
    ntterm(CONNECT_ERR);
    }
```

4.3 NTM Service Call Parameters

AP_NAME

The parameter AP-NAME contains a 10-character alphanumeric. To prevent conflicts with system components, application names starting with "NT" are reserved.

```
C       char ap_name [10];
```

BINARY_NATIVE_FLAG

The parameter BINARY-NATIVE-FLAG contains a code which specifies the generic type of data contained in the data portion of a message. Binary (B) indicates that the data is in the host machine's internal representation form whereas native (N) indicates that the data is character data represented by the host machine's character code (ASCII, EBCDIC, etc.).

```
C       char binary_native_flag;
```

DATA

The parameter DATA contains an alphanumeric which represents the data portion of the message currently being processed.

```
C       char data [data_length];
```

DATA_LENGTH

The parameter DATA_LENGTH contains a numeric value which specifies the length of the data portion of the message currently being processed. Maximum size allowed is 65535.

```
C       int data_length;
```

DESTINATION

The parameter DESTINATION contains a 10-character alphanumeric that is used to specify the destination AP for the message currently being processed. See the description of AP-NAME for more detail.

```
C       char destination [10];
```

ERROR_CODE

The parameter ERROR_CODE contains the value associated with the error that triggers a call to "NTSERR".

```
C       char error_code [5];
```

ERROR-DESCRIPTION

The parameter ERROR_DESCRIPTION contains information about the error triggering a call to "NTSERR". The content of this parameter is defined by the calling AP.

C char error_description [72];

HOST-NAME

The parameter HOST_NAME contains an 8-character alphanumeric that is used to specify the "physical" name of the host computer system that the caller is on.

C char host_name [8];

LOGICAL_CHANNEL

The parameter LOGICAL_CHANNEL contains a value which is used by the APs and the NTM to manage communication paths between APs (see Section 3.1.5). A value must be supplied on a SEND call when the AP wants to pair the send with a response. A blank logical channel is not valid.

C char logical_channel [3];

MESSAGE_TYPE

The parameter MESSAGE_TYPE contains a code which specifies the message type of the message currently being processed. A blank MESSAGE-TYPE is not valid.

C char message_type [2];

OPTIONS

The parameter OPTIONS specifies the characteristics of the transfer and the files.

C char options[80];

REQNO

The parameter REQNO contains a unique identifier for a file transfer request.

C char reqno[3];

ROLE_NAME

The parameter ROLE_NAME contains a 10-character alphanumeric which identifies the role under which a user is logged on.

C char role_name [10];

SEVERITY_LEVEL

The parameter contains a code specifying the level of the error that triggered a call to "NTSERR". Acceptable values are F,E,W, I.

C char severity_level;

SOURCE

The parameter SOURCE contains a 10-alphanumeric that is used to specify the source AP for a call to NTRECV with a source match. See the description of AP_NAME for more detail.

C char source [10];

SYSTEM_STATE

The parameter SYSTEM_STATE contains a code which indicates the system state of the NTM at the time "NTINIT" is called.

C char system_state;

TERMINAL_ID

The parameter TERMINAL_ID identifies the terminal where a given user is logged on.

C char terminal_id [8];

TERMINATION_STATUS

The parameter TERMINATION_STATUS contains a code which indicates the specific condition under which an AP is terminating.

C char termination_status;

TEST_STATUS

The parameter TEST_STATUS contains a code which indicates whether the AP wishes to receive any asynchronous error message, fatal errors only, or not at all. Its description in COBOL is

C char test_status;

TIMEOUT_VALUE

The parameter TIMEOUT_VALUE contains a numeric value which represents a time value in seconds which is used in conjunction with a paired message request. Valid values for timeout are 0 through 86400.

C int timeout_value;

USER_NAME

The parameter USER_NAME contains an 8-character alphanumeric value which identifies a specific user.

C char user_name [8];

WAIT_FLAG

The parameter WAIT_FLAG contains a code which indicates whether or not the message currently being processed should have a wait associated with it.

C char wait_flag;

A value of "0" indicates no waiting in requested and a value of "1" indicates the API should wait.

4.4 Return Code Values

NAME

ntmapi.h -- NTM APplication Interface return codes

Description

Return codes for verion 2.

```
#define NTM_SUCCESS                     0

#define NTM_NO_MESSAGE                  1
#define NTM_REPLY_REQUIRED_MESSAGE      2
#define NTM_TIME_OUT_MESSAGE            3
#define NTM_TEST_MODE_MESSAGE           4
#define NTM_HOST_UP                     10
#define NTM_HOST_DOWN                   11
#define NTM_TEST_MODE_ON                20
#define NTM_TEST_MODE_OFF               21
#define NTM_TEST_MODE_FATAL             22

#define NTM_FAILURE                     -1
#define NTM_BUFFER_TOO_SMALL            -2
#define NTM_INVALID_DATA_LENGTH         -3
#define NTM_INVALID_DATA_TYPE           -4
#define NTM_INVALID_DESGINATION         -5
#define NTM_INVALID_LOGICAL_CHANNEL     -6
#define NTM_INVALID_MESSAGE_TYPE        -7
#define NTM_INVALID_SOURCE              -8
#define NTM_INVALID_TIME                -9
#define NTM_INVALID_TIMEOUT_REQUEST     -10
#define NTM_INVALID_TRG_COND            -11
#define NTM_NOT_AUTHORIZED              -12
#define NTM_AP_NOT_FOUND                -13
#define NTM_HOST_NOT_KNOWN              -14
#define NTM_NOT_CONNECTED               -15
#define NTM_ALREADY_CONNECTED           -16
#define NTM_USER_NOT_LOGGED_ON          -17
#define NTM_USER_ALREADY_LOGGED_ON      -18
#define NTM_FILE_SOURCE_NAME_INVALID    -100
```

```
#define NTM_FILE_SOURCE_HOST_INVALID    -102
#define NTM_FILE_DEST_HOST_INVALID      -103
#define NTM_FILE_OPTIONS_INVALID        -104
#define NTM_RESOURCES_NOT_AVAILABLE     -900
#define NTM_RESOURCE_UNAVAIL_HOST       -901
#define NTM_RESOURCE_UNAVAIL_APC        -902
#define NTM_RESOURCE_UNAVAIL_LINK       -903
#define NTM_RESOURCE_UNAVAIL_AP         -904
#define NTM_RESOURCE_UNAVAIL_SYS        -905
#define NTM_RESOURCE_UNAVAIL_MEMORY     -906

#define NTM_INIT_RECOVERY               '1'
#define NTM_INIT_NTM_START              '2'
#define NTM_INIT_NORMAL                 '3'
#define NTM_INIT_FIRST-RUN              '4'

#define NTM_TERM_NORMAL                 '1'
#define NTM_TERM_SHUTDOWN               '2'
#define NTM_TERM_ABORT                  '3'
#define NTM_TERM_EXCEPTION              '4'
#define NTM_TERM_INIT_FAIL              '5'
#define NTM_TERM_NORMAL_RET             '6'
#define NTM_TERM_SHUTDOWN_RET           '7'
#define NTM_TERM_ABORT_RET              '8'
#define NTM_TERM_EXCEPTION_RET          '9'
```

SECTION 5

NTM VERSION 1.1 COMPATIBILITY INTERFACE

5.1 Services Available to All NTM Applications

This section describes the following services which are available to all NTM applications.

CHKMSG
GETUSR
HSTATS
INITAL
ISEND
NSEND
QSEND
RCV
SIGABT
SIGERR
TRMNAT
TSTMOD
WHTHST

The use of each service is explained in detail in the following sections. This includes a description of the service's purpose, return parameter values, the calling parameters' formats, and example calls. Note that Str and Int in the format mean character string and long integer, respectively.

A detailed description of the calling parameters with specific COBOL and C definitions are contained in Section 5.3. The values of the service returns are defined in Section 5.4.

5.1.1 CHKMSG

Check for the arrival of messages in the AP's mailbox.

Calling Sequence

```
CALL CHKMSG  (LOGICAL-CHANNEL,
              SOURCE,
              DATA-LENGTH
              RET-CODE)
```

Description

CHKMSG can be used to determine whether either any message or a specific message has arrived at the AP. The size of the message that would be received is returned in the DATA-LENGTH parameter. The message is retrieved with the "RCV" service. CHKMSG does not deliver messages to the AP.

- To check for any message, leave the logical channel and source arguments blank. If more than one message is pending, the channel and source of the first message in the buffer will be returned.
- To check for a message from a specific source on any channel, specify the source argument and leave the channel argument blank. If more than one message has arrived from the specified source, the channel of the first message in the buffer from the specified source will be returned.
- To check for a message from a specific source on a specific channel, specify both source and logical channel If any messages have arrived from this source on this channel, the return code will indicate NTM-SUCCESS.
- To check for hot messages from the NTM (such as Shutdown Pending, Cancel Shutdown, or Shutdown), the message source must be specified as "NTM ". The seven following blanks are required by the NTM naming conventions.

Inputs or Outputs

LOGICAL-CHANNEL	str (3)
SOURCE	str (10)

Outputs

DATA-LENGTH	int
RET-CODE	str (5)

RET-CODE Values

NTM-SUCCESS
The specified (or any, depending on values in the call parameters) message was found

NTM-NO-MESSAGE
The specified message was not found - or - if any, no message was found.

NTM-FAILURE
An error has occurred within the CHKMSG routine.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

COBOL

1. To check for any cold message:

```
MOVE SPACES TO MSG-SOURCE.
MOVE SPACES TO LOGICAL-CHANNEL.
CALL "CHKMSG" USING LOGICAL CHANNEL.
                    MSG-SOURCE,
                    DATA-LENGTH,
                    RET-CODE

IF NTM-SUCCESS
   PERFORM RECEIVE-MESSAGE
ELSE
   NEXT SENTENCE
```

2. To check for an NTM message:

```
MOVE SPACES TO LOGICAL-CHANNEL.
MOVE "NTM" to MSG-SOURCE.
CALL "CHKMSG" USING LOGICAL-CHANNEL,
                    MSG-SOURCE,
                    DATA-LENGTH,
                    RET-CODE
IF NTM-SUCCESS
   PERFORM RCV-UNSOL-MSG
ELSE
   NEXT SENTENCE.
```

C LANGUAGE

1. To check for any cold message:

```
memset (source, ' ', sizeof (source));
memset (chan, ' ', sizeof (chan));
chkmsg (source, chan, length, rcode);
if (memcmp (rcode, NTM_SUCCESS, sizeof (rcode)==0)
   rcv_msg();
```

2. To check for an NTM Message

```
memcpy (source, "NTM      ", sizeof (source));
memset (chan, ' ', sizeof (chan));
chkmsg (source, chan, length, rcode);
if (memcmp (rcode, NTM_SUCCESS, sizeof (rcode))==0
   rec_unsol_msg();
```

5.1.2 GETUSR

Determine the AP Name and user name of the original node or source in an AP chain. The User Logon information is determined only when the original source AP has performed a call to Logon.

Calling Sequence

```
CALL GETUSR   (AP-NAME,
               LOGICAL-CHANNEL,
               USER-NAME,
               ROLE-NAME,
               TERMINAL-ID,
               RETURN-CODE)
```

Description

The parameters that GETUSR returns to the caller are the current user name, AP name and logical channel associated with its original source. This call allows any AP in a chain to determine its originating source and chain communication channel. If the originating source is a user at a terminal, a user-name and the name of the associated AP, are returned.

If the originating source is an AP (not a terminal user), then the user Logon data will be blank on return. Only the AP-Name and Logical-Channel will have non-blank values.

Inputs

None

Outputs

AP-NAME	str (10)
LOGICAL-CHANNEL	str (3)
USER-NAME	str (8)
ROLE-NAME	str (10)
TERMINAL-ID	str (8)
RETURN-CODE	str (5)

RETURN-CODE Values

NTM-SUCCESS
The GETUSR service has successfully obtained all of the requested data.

NTM-F ' 'LURE
The GETUSR service was not able to obtain the requested data. This would be due to an error in accessing the LOGON Table.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

COBOL

Using Call "GETUSR" to obtain the values needed to send a message to the AP at the head of the chain.

```
CALL "GETUSR" USING AP-NAME,
                    ORIG-CHANNEL,
                    USER-NAME,
                    ROLE-NAME,
                    TERM-ID,
                    RET-CODE.
IF NTM-SUCCESS
    NEXT SENTENCE
ELSE
    MOVE "8" TO GOOF-CODE
    PERFORM GOOF-IN-PROGRAM.
MOVE "XTSAP5 HAS DONE ITS JOB." TO LAST-MSG.
MOVE AP-NAME TO MSG-DESTINATION.
MOVE ORIG-CHANNEL TO LOGICAL-CHANNEL.
MOVE 25 TO DATA-LENGTH-SEND.
MOVE "DM" TO MESSAGE-TYPE-SEND.
MOVE ZERO TO TIMEOUT-VALUE.
MOVE LAST-MSG TO DATA-SEND.
CALL "NTNSND" USING MSG-DESTINATION,
                    LOGICAL-CHANNEL,
                    TIMEOUT-VALUE,
                    BINARY-NATIVE-FLAG,
                    MESSAGE-TYPE-SEND,
                    DATA-LENGTH-SEND,
                    DATA-SEND,
                    RET-CODE.
IF NTM-SUCCESS
    PERFORM FINISH-PROGRAM
ELSE
    MOVE "4" TO GOOF-CODE
    PERFORM GOOF-IN-PROGRAM.
```

C LANGUAGE

```
getusr (apname, chan, user, role, term, rcode);
if (memcmp (rcode, NTM_SUCCESS, sizeof (rcode))==0
    {
    strcpy (data, "xtsap5 has done its job");
    nsend (apname, chan, & 0, "N", "DM", & length, data, rcode);
    if (memcmp (rcode, NTM-SUCCESS, size of (rcode))!=0
        {
        goof-in-program ("4");
        }
    }
else
    {
    goof in program ("8");
    }
```

5.1.3 HSTATS

Get the status of a specified HOST.

Calling Sequence

```
CALL HSTATS    (HOST-NAME,
                RET-CODE)
```

Description

HSTATS returns to the caller a status code containing relevant information about the HOST which was specified in the request.

Inputs

HOST-NAME	str (8)

Outputs

RET-CODE	str (5)

RET-CODE Values

NTM-HOST-UP
The specified host is active.

NTM-HOST-DOWN
The specified host is not active.

NTM-HOST-NOT-KNOWN
The specified host is not part of the NTM configuration.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

COBOL

```
MOVE "IBM" to HOST-NAME.
CALL "HSTATS" USING HOST-NAME,
                    NTM-RET-CODE.
IF NOT NTM-HOST-UP
    MOVE "01" TO GOOF-CODE
    PERFORM SERVICE-ERROR
ELSE
    NEXT SENTENCE.
```

C LANGUAGE

```
static char host[9] = "IBM";
char rcode[5];

hstats(host, rcode);
if (memcmp(rcode, NTM_HOST_UP, size of (rcode)) != 0)
    printf("Host unavailable");
```

5.1.4 INITAL

Provide initiation services for an AP.

Calling Sequence

```
CALL INITAL   (BUFFER,
               BUFFER-SIZE,
               SYSTEM-STATE,
               RET-CODE)
```

Description

INITAL is the routine called by an AP to request that the AP interface perform the necessary initialization to allow the AP to execute and communicate with the NTM environment.

Inputs

BUFFER	str (*)
BUFFER-SIZE	int (1-65535)

Outputs

SYSTEM-STATE	str (1)
RET-CODE	str (5)

RET-CODE Values

NTM-SUCCESS
The AP has successfully connected with the NTM.

NTM-FAILURE
The AP did not connect with the NTM.

NTM-RESOURCES-NOT-AVAILABLE
The connection was not made to the NTM. As this is a temporary condition, the initial call may be tried again.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

SYSTEM-STATE Values

NTM-INIT-NORMAL
The NTM is operating normally.

NTM-INIT-FIRST-RUN
Identifies the first run of the AP after an event that is significant to the AP.

Examples

COBOL

An AP that does no recovery, but does special first-run processing.

```
DATA DIVISION.
WORKING-STORAGE SECTION.
COPY SRVRET OF INCLIB.
01 BUFFER PIC X(4096).
01 BUFFER-SIZE PIC 9(5) COMP VALUE 4096.
PROCEDURE DIVISION.
START PROGRAM.
    CALL "INITAL" USING BUFFER,
                        BUFFER-SIZE,
                        SYSTEM-STATE,
                        RET-CODE.
IF NTM-SUCCESS
    IF NTM-INIT-FIRST-RUN
            PERFORM NTINIT-CODE
            PERFORM RUN-CODE
        ELSE
            PERFORM RUN-CODE
ELSE
    PERFORM TERMINATION-CODE.
INITIAL-CODE.
RUN-CODE.
TERMINATION-CODE.
```

C LANGUAGE

```
static char ntmbuf[128];
static int ntmsiz = 129;
char systate;
char rcode[5];

initial(ntmbuf, &ntmsiz, &systate, rcode);
if (memcmp(rcode, NTM_SUCCESS, size of (rcode)) != 0)
    {
    printf ("Unable to connect to NTM");
    trmnat(CONNECT_ERR);
    }
```

5.1.5 ISEND

Send a message that requests the specific initiation of a new instance of an AP. Data for the new destination AP instance may be included in the message.

Calling Sequence

```
CALL ISEND      (DESTINATION,
                 LOGICAL-CHANNEL,
                 TIMEOUT-VALUE,
                 BINARY-NATIVE-FLAG,
                 MESSAGE-TYPE,
                 DATA-LENGTH,
                 DATA,
                 RET-CODE)
```

Description

ISEND is used to specifically request the initiation of a new instance of a destination AP Except for this feature, the services provided to an AP by this call are the same as NSEND. This call is intended to support complex APs that must manage communications between multiple instances of the same AP. The ISEND user must specify different channel indicators to manage the communication between multiple instances of the same AP. The ISEND call may be used with or without data. (Also see Programming Guidelines for Queue Server Applications.)

The ISEND call must be used when starting APs having the characteristic of requiring a specific initiation. For APs having no restriction on initiation, the use of call ISEND serves to guarantee the initiation of a new instance.

Inputs

DESTINATION	str (10)
LOGICAL-CHANNEL	str (3)
TIMEOUT-VALUE	int
BINARY-NATIVE-FLAG	str (1)
MESSAGE-TYPE	str (2)
DATA-LENGTH	int
DATA	str (*)

LOGICAL-CHANNEL is optional. (Field should be blank if no channel is required.) TIMEOUT-VALUE should be set to zero if pairing support is not required. A non-zero timeout value will activate the message pairing processing. The Destination argument must include the directory prefix.

Outputs

RET-CODE	str (5)

RET-CODE Values

NTM-SUCCESS
The message sent under any of the SEND calls has been accepted by the NTM.

NTM-NOT-AUTHORIZED
The message failed the AUTHORIZED authorized check performed by the NTM. The source is not authorized to send a message of the given type to the destination.

NTM-INVALID-MESSAGE-TYPE
The message type argument is blank.

NTM-INVALID-DESTINATION
The destination AP name in the calling argument was not found in the NTM tables.

NTM-INVALID-DATA-LENGTH
The data length argument is blank.

NTM-INVALID-DATA-TYPE
The given binary-native flag is not valid. The value must be either "B" or "N".

NTM-RESOURCES-NOT-AVAILABLE
The resources needed by the NTM to process the message are not available.

NTM-INVALID-TIMEOUT-REQUEST
The given timeout value is not valid.

NTM-FAILURE
The NTM has rejected the message for reasons beyond the control of the source AP.

NTM-INVALID-SOURCE
The source AP name cannot be found in the NTM tables.

NTM-INVALID-LOGICAL-CHANNEL
The logical channel specified does not identify a valid queue server connection.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

COBOL

```
MOVE "TSTESTAPPL" TO MSG-DESTINATION.
MOVE "002" TO LOGICAL-CHANNEL.
MOVE "IR" TO MESSAGE-TYPE-SEND.
MOVE 25 TO DATA-LENGTH-SEND.
MOVE "SEND PART INVENTORY DATA." TO DATA-SEND.
MOVE ZERO TO TIMEOUT-VALUE.
CALL "ISEND" USING MSG-DESTINATION,
                   LOGICAL-CHANNEL,
                   TIMEOUT-VALUE,
                   BINARY-NATIVE-FLAG,
                   MESSAGE-TYPE-SEND,
                   DATA-LENGTH-SEND,
                   DATA-SEND,
                   RET-CODE.
IF NTM-SUCCESS
    DISPLAY "MESSAGE IS ON ITS WAY."
ELSE
    MOVE "05" TO GOOF-CODE
    PERFORM SERVICE-ERROR.
```

C LANGUAGE

```
char destxfer[12];
char req_no[4];
static int timeout = 0;
int cmdlen;
char command[256];
char rcode[5];

isend(destxfer, req_no, &timeout, "N", "MT", &cmdlen, command, rcode);
if (memcmp(rcode, NTM_SUCCESS, size of (rocde)) == 0)
    printf ("Message is on its way");
else
    printf ("isend return code =%.5s", rcode);
```

5.1.6 NSEND

Send a message through the NTM.

Calling Sequence

```
CALL NSEND    (DESTINATION,
               LOGICAL-CHANNEL,
               TIMEOUT-VALUE,
               BINARY-NATIVE-FLAG,
               MESSAGE-TYPE,
               DATA-LENGTH,
               DATA,
               RET-CODE)
```

Description

NSEND is used to send a message that does not require special NTM handling from an AP to any authorized destination.

Inputs

DESTINATION	str (10)
LOGICAL-CHANNEL	str (3)
TIMEOUT-VALUE	int
BINARY-NATIVE-FLAG	str (1)
MESSAGE-TYPE	str (2)
DATA-LENGTH	int
DATA	str (*)

LOGICAL-CHANNEL can be blank if a specific channel is not required. TIMEOUT-VALUE must be zero if pairing support is not required. If a non-zero timeout value is specified, the AP must specify a non-blank LOGICAL-CHANNEL in order to pair the responses or timeout message. The Destination argument must specify the directory prefix See also Programming Guidelines for Queue Server Applications.

Outputs

RET-CODE	str (5)

RET-CODE Values

NTM-SUCCESS
The message sent under any of the SEND calls has been accepted by the NTM.

NTM-NOT-AUTHORIZED
The message failed the AUTHORIZED authorized check performed by the NTM. The source is not authorized to send a message of the given type to the destination.

NTM-INVALID-MESSAGE-TYPE
The message type argument is blank.

NTM-INVALID-DESTINATION
The destination AP name in the calling argument was not found in the NTM tables.

NTM-INVALID-DATA-LENGTH
The data length argument is blank.

NTM-INVALID-DATA-TYPE
The given binary-native flag is not valid. The value must be either "B" or "N".

NTM-RESOURCES-NOT-AVAILABLE
The resources needed by the NTM to process the message are not available.

NTM-INVALID-TIMEOUT-REQUEST
The given timeout value is not valid.

NTM-FAILURE
The NTM has rejected the message for reasons beyond the control of the source AP.

NTM-INVALID-SOURCE
The source AP name cannot be found in the NTM tables.

NTM-INVALID-LOGICAL-CHANNEL
The logical channel specified does not identify a valid queue server connection.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

COBOL

1. To send a paired message. Note: The value of Binary-Native-Flag has been set in the DATA DIVISION.

```
MOVE "TSTESTAPPL" TO MSG-DESTINATION.
MOVE "003" TO LOGICAL-CHANNEL.
MOVE "ID" TO MESSAGE-TYPE-SEND.
MOVE "DETERMINE THE LOCATION OF PART X AND MOVE ALL TO" "
POINT B." TO DATA-SEND.
MOVE 57 TO DATA-LENGTH-SEND.
MOVE 240 TO TIMEOUT-VALUE.
CALL "NSEND" USING MSG-DESTINATION,
                   LOGICAL-CHANNEL,
                   TIMEOUT-VALUE,
                   BINARY-NATIVE-FLAG,
                   MESSAGE-TYPE-SEND,
                   DATA-LENGTH-SEND,
                   DATA-SEND,
                   RET-CODE.

IF NTM-SUCCESS
    DISPLAY "MESSAGE IS ON ITS WAY."
ELSE
    MOVE "06" TO GOOF-CODE
    PERFORM SERVICE-ERROR.
```

2. To send an unpaired message:

```
MOVE "TSTESTAPPL" TO MSG-DESTINATION.
MOVE "005" TO LOGICAL-CHANNEL.
MOVE "ID" TO MESSAGE-TYPE-SEND.
MOVE "HELLO " TO DATA-SEND.
MOVE 5 TO DATA-LENGTH-SEND.
MOVE ZERO TO TIMEOUT-VALUE.
CALL "NSEND" USING MSG-DESTINATION,
                   LOGICAL-CHANNEL,
                   TIMEOUT-VALUE,
                   BINARY-NATIVE-FLAG,
                   MESSAGE-TYPE-SEND,
                   DATA-SEND,
                   RET-CODE.
IF NTM-SUCCESS
    DISPLAY "MESSAGE IS ON ITS WAY."
ELSE
    MOVE "06" TO GOOF-CODE
    PERFORM SERVICE-ERROR.
```

C LANGUAGE

```
char rmsgtyp[2];
char rcode[5];
static int zerotime = 0;
int lngth;

nsend(destination,log-chan,&zerotime,ASCII,rmsgtyp,&lngth,buff,rcode);
if (memcmp(rcode,NTM_SUCCESS,size of (rocde))  != 0)
    printf ("Message not sent");
else
    printf ("NTISND return code =%.5s", rcode);
```

5.1.7 QSEND

Used by a queue server AP to send a message which terminates a connection between the queue server and its parent AP.

Calling Sequence

```
CALL QSEND    (DESTINATION,
               LOGICAL-CHANNEL,
               TIMEOUT-VALUE,
               BINARY-NATIVE-FLAG,
               MESSAGE-TYPE,
               DATA-LENGTH,
               DATA,
               RET-CODE)
```

Description

QSEND is the service used by a queue server AP to send the final message in a sequence of messages with an AP that has established a connection with the queue server (via NTISND). The "QSEND" message terminates the connection between the queue server and its parent.

Inputs

DESTINATION	str (10)
LOGICAL-CHANNEL	str (3)
TIMEOUT-VALUE	int
BINARY-NATIVE-FLAG	str (1)
MESSAGE-TYPE	str (2)
DATA-LENGTH	int
DATA	str (*)

Outputs

RET-CODE	str (5)

RET-CODE Values

NTM-SUCCESS
The message sent under any of the SEND calls has been accepted by the NTM.

NTM-NOT-AUTHORIZED
The message failed the AUTHORIZED authorized check performed by the NTM. The source is not authorized to send a message of the given type to the destination.

NTM-INVALID-MESSAGE-TYPE
The message type argument is blank.

NTM-INVALID ` TINATION
The destination A. name in the calling argument was not found in the NTM tables.

NTM-INVALID-DATA-LENGTH
The data length argument is blank.

NTM-INVALID-DATA-TYPE
The given binary-native flag is not valid. The value must be either "B" or "N".

NTM-RESOURCES-NOT-AVAILABLE
The resources needed by the NTM to process the message are not available.

NTM-INVALID-TIMEOUT-REQUEST
The given timeout value is not valid.

NTM-FAILURE
The NTM has rejected the message for reasons beyond the control of the source AP.

NTM-INVALID-SOURCE
The source AP name cannot be found in the NTM tables.

NTM-INVALID-LOGICAL-CHANNEL
The logical channel specified does not identify a valid queue server connection

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

COBOL

```
MOVE "TSTESTAPPL" TO MSG-DESTINATION.
MOVE "003" TO LOGICAL-CHANNEL.
MOVE "ID" TO MESSAGE-TYPE-SEND
MOVE "TESTING QSEND" TO DATA-SEND.
MOVE 13 TO DATA-LENGTH-SEND.
MOVE ZERO TO TIMEOUT-VALUE.
CALL "QSEND" USING MSG-DESTINATION,
LOGICAL-CHANNEL,
TIMEOUT-VALUE,
BINARY-NATIVE-FLAG,
MESSAGE-TYPE-SEND,
DATA-LENGTH-SEND,
DATA-SEND,
NTM-RET-CODE.
IF NTM-SUCCESS
    NEXT SENTENCE
ELSE
    MOVE "4" TO GOOF-CODE
    PERFORM GOOF-IN-PROGRAM.
```

C LANGUAGE

```
char destxfer[12];
char req_no[4];
static int timeout = 0;
int cmdlen;
char command[256];
char rcode[5];

qsend(destxfer, req_no, &timeout, "N", "MT", &cmdlen, command, rcode);
if (memcmp(rcode, NTM_SUCCESS, size of (rcode)) == 0)
    printf ("Message is on its way");
else
    printf ("QSEND return code = %.5s", rcode);
```

5.1.8 RCV

Receive a message through the NTM.

Calling Sequence

```
CALL RCV  (LOGICAL-CHANNEL,
           WAIT-FLAG,
           SOURCE,
           MESSAGE-TYPE,
           BUFFER-SIZE,
           DATA-LENGTH,
           DATA,
           RET-CODE)
```

Description

RCV is used to receive a message, of any type, from any authorized source, including the NTM itself, via the services provided by the NTM. See also Programming Guidelines for Queue Server Applications.

Inputs

WAIT-FLAG	str (1)
BUFFER-SIZE	int

Outputs

DATA-LENGTH	int
DATA	str (*)
RET-CODE	str (5)

Inputs/Outputs

MESSAGE-TYPE	str (2)
LOGICAL-CHANNEL	str (3)
SOURCE	str (10)

RET-CODE Values

NTM-SUCCESS
A message has been received.

NTM-REPLY-REQUIRED-MESSAGE
A paired message has been received.

NTM-NO-MESSAGE
No message was found in the AP's mailbox.

NTM-TIME-OUT-MESSAGE
A message time-out notification has arrived from the NTM.

NTM-TEST-MODE-MESSAGES
The message received is from an AP currently operating in test mode.

NTM-FAILURE
The NTM cannot access the Inter-Process Communication facilities.

NTM-BUFFER-TOO-SMALL
The buffer size specified is not large enough to hold the received message.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

NOTES

1. SOURCE, LOGICAL-CHANNEL, and/or MESSAGE-TYPE can be specified if selective message receiving is required. If source message type and logical-channel are blank, the first message in the buffer will be returned with its source, message type, and logical-channel. On a source match, the directory prefix must be specified.

2. If the AP wishes to specify the NTM for a source match, the source parameter must be entered as "NTM" on the call.

3. If the RCV service is called after a call to CHKMSG, the source and channel parameter should be those returned on the CHKMSG call.

4. If the AP expects to receive paired messages it must use the condition "IF RCV-REPLY-REQUIRED-MESSAGE" to test the RCV call return.

5. If "RCV" is called with no wait set, the AP must be able to deal with the possibility of a "NTM-NO-MESSAGE" return.

Examples

COBOL

1. RCV with no wait - to retrieve any message.

```
MOVE SPACES TO MSG-SOURCE.
MOVE SPACES TO LOGICAL-CHANNEL.
MOVE SPACES TO DATA-RCV.
MOVE "0" TO WAIT-FLAG.
CALL "RCV" USING LOGICAL-CHANNEL,
                 WAIT-FLAG,
                 MSG-SOURCE,
                 MESSAGE-TYPE-RCV,
                 BUFFER-SIZE,
                 DATA-LENGTH-RCV,
                 DATA-RCV,
                 RET-CODE,
IF NTM-SUCCESS
    ADD 1 TO RCV-COUNT
ELSE IF NTM-NO-MESSAGE
    DISPLAY "NO MESSAGE YET - HANG IN THERE!"
ELSE
    MOVE "07" TO GOOF-CODE
    PERFORM SERVICE-ERROR.
```

2. RCV with a wait - The AP expects to receive paired messages.

```
MOVE "1" TO WAIT-FLAG.
MOVE SPACES TO LOGICAL-CHANNEL.
CALL "RCV" USING LOGICAL-CHANNEL,
                 WAIT-FLAG,
                 MSG-SOURCE,
                 MESSAGE-TYPE-RCV,
                 BUFFER-SIZE,
                 DATA-LENGTH-RCV,
                 RET-CODE,
                 MESSAGE-SERIAL-NUMBER.
IF NTM-SUCCESS
OR NTM-REPLY-REQUIRED MESSAGE
    NEXT SENTENCE
ELSE
    MOVE "1" TO GOOF-CODE
    PERFORM GOOF-IN-PROGRAM.
```

3. RCV with a wait - The AP is looking for a specific message.

```
MOVE "1" TO WAIT-FLAG.
MOVE "TSTESTAPPL" TO MSG-SOURCE.
MOVE "001" TO LOGICAL-CHANNEL.
CALL "RCV" USING LOGICAL-CHANNEL,
                 WAIT-FLAG,
                 MSG-SOURCE,
                 MESSAGE-TYPE-RCV,
                 BUFFER-SIZE,
                 DATA-LENGTH-RCV,
                 DATA-RCV,
                 RET-CODE,
IF NTM-SUCCESS OR NTM-REPLY-REQUIRED-MESSAGE
    NEXT SENTENCE
ELSE
    MOVE "1" TO GOOF-CODE
    PERFORM GOOF-IN-PROGRAM.
```

C LANGUAGE

```
char chan[4];
char destxfer[12];
char msgtyp[2];
static int bufsiz = 256;
int msglen;
char msgbuf[256];
char rcode[5];

rcv(chan, "1", destxfer, msgtyp, &bufsiz, &msglen, msgbuf, rcode);
if (memcmp(rcode, NTM-SUCCESS, size of (rcode)) != 0)
    goto error;
```

5.1.9 SIGABT

Signal to the NTM that an AP is to be aborted.

Calling Sequence

```
CALL SIGABT    (AP-NAME,
                LOGICAL-CHANNEL,
                RET-CODE)
```

Description

SIGABT allows the calling AP to indicate to the NTM that the AP wishes to abort another AP which it (the caller) has spawned.

Inputs

AP-NAME	str (10)
LOGICAL-CHANNEL	str (3)

Outputs

RET-CODE	str (5)

RET-CODE Values

NTM-SUCCESS
The message to abort a child AP has been accepted for processing by the NTM.

NTM-FAILURE
The message to abort an AP has not been accepted by the NTM.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Notes

These code values only signify the acceptance of the abort message by the NTM. If the AP requires a specific acknowledgement of the actual abort, it must indicate this requirement when it defines its characteristics to the NTM system Administrator as described in the following paragraph.

An Application Program (AP) can request the NTM to abort another AP that the caller has spawned by using the NTM SIGABT call. The Return Code from the NTM will only indicate whether the SIGABT message has been accepted or rejected for processing by the NTM, not whether the Application Program has been aborted. The Application Program can determine if the AP to be aborted has terminated by waiting for NTM "Application End" messages after the call to SIGABT has been made. Note that these messages will be delivered to the requesting AP if the following conditions exist:

1. The requesting AP has been given characteristics in the application Characteristics Table (APTTBL) of

On Child Abort	Send Message	"1"
On Child Termination	Send Message	"1"
On Child Shutdown	Send Message	"1"

2. The "to be" AP has been given the characteristics in the Application Characteristics Table (APTTBL) of

On Abort	Send Abort	"2"	or
On Abort	Abort	"3"	

Note that if the "to be aborted" AP has been given the On Abort Characteristic of Run to Completion (indicated by a "1" in the APTTBL) then the AP will **not** be aborted and will not receive any notification of the SIGABT request.

Examples

COBOL

```
MOVE "ABADAPPLIC"  TO AP-NAME.
MOVE "001" TO LOGICAL-CHANNEL.
CALL "SIGABT" USING AP-NAME,
                    LOGICAL-CHANNEL,
                    NTM-RET-CODE.
IF NTM-SUCCESS
    DISPLAY "ABORT SIGNAL ACCEPTED BY NTM."
ELSE
    DISPLAY "ABORT SIGNAL NOT ACCEPTED.".
DISPLAY " ".
```

C LANGUAGE

```
static char chan[4];

sigabt(dest, chan, rcode);
if (memcmp(rcode, NTM_SUCCESS, size of (rcode)) != 0)
    printf ("Abort signal not accepted.")
else
    printf ("Abort signal accepted by the NTM.");
```

5.1.10 SIGERR

Allows an AP to signal the NTM and its original source when a non-fatal error occurs.

Calling Sequence

```
CALL SIGERR     (ERROR-CODE,
                 SEVERITY-LEVEL,
                 ERROR-DESCRIPTION,
                 RET-CODE)
```

Description

SIGERR accepts error data from the calling AP and forwards that data to both the head of the calling AP's chain and the Monitor AP.

Inputs

ERROR-CODE	str (5)
SEVERITY-LEVEL	str (1)
ERROR-DESCRIPTION	str (72)

Outputs

RET-CODE	str (5)

RET-CODE Values

NTM-SUCCESS
The SIGERR message has been sent successfully.

NTM-FAILURE
The SIGERR message could not be sent.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

COBOL

```
MOVE "07001" TO ERROR-CODE.
MOVE "I" TO SEVERITY-LEVEL.
CALL "SIGERR" USING ERROR-CODE,
                    SEVERITY-LEVEL,
                    ERROR-DESCRIPTION,
                    NTM-RET-CODE.
IF NTM-SUCCESS
    DISPLAY "MESSAGE SENT SUCCESSFULLY"
ELSE MOVE "01" TO GOOF-CODE
    PERFORM SERVICE-ERROR.
```

C LANGUAGE

```
static char code[6] = "99925";
static char sev = 'W';
static char msg[61] = "Example of sigerr call";
char rcode[5];

sigerr(code, &sev, msg, rcode);
if (memcmp(rocde, NTM_SUCCESS, size of (rcode)) != 0)
    printf ("SIGERR failed");
```

5.1.11 TRMNAT

Signal AP termination status to the NTM.

Calling Sequence

CALL TRMNAT (TERMINATION-STATUS)

Description

TRMNAT performs termination processing for an executing AP and allows the AP to provide a status code to the NTM specifying the termination conditions.

Inputs

TERMINATION-STATUS str (1)

Outputs

None

TERMINATION-STATUS Values

TRMNAT-NORMAL
The AP has TERMINATION terminated normally.

TRMNAT-SHUTDOWN
The AP has COMPLETE completed its shutdown processing.

TRMNAT-ABORTED
The AP has terminated as the result of a soft abort command.

TRMNAT-EXCEPTION
The AP has terminated as the result of an internal (to the AP) exception condition.

TRMNAT-ON-BAD-INIT
The AP has terminated due to an error incurred when trying to connect to the NTM.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

NOTES

1. The Termination Status values defined above must be used in this call. These values are used by the NTM to direct the clean-up procedures executed upon the AP's termination

2. The call "TRMNAT" with TERMINATION-STATUS set to TRMNAT-ON BAD-INIT must be used if the AP did not successfully connect to the NTM in call "NTINIT."

3. Any value other than those above will be treated as "1".

Examples

COBOL

```
IF NOT-NTM-INIT-NORMAL
    MOVE TRMNAT-ON-BAD-INIT TO TERMINATION-STATUS
ELSE
    MOVE TRMNAT-NORMAL-TERMINATION TO TERMINATION-STATUS
CALL "TRMNAT" USING TERMINATON-STATUS.
```

C LANGUAGE

```
trmnat (PROGRAM_ERR);
```

5.1.12 TSTMOD

Switch NTM message test mode on or off. When test mode is on, the calling AP will be able to receive asynchronous error messages. When test mode is off, these error messages will be discarded.

Calling Sequence

CALL TSTMOD (TEST-STATUS,
RET-CODE)

Description

TSTMOD is used by a calling AP to indicate to the NTM whether or not it wishes to receive asynchronous error messages.

Inputs

TEST-STATUS str (1)

Values:

"0" = Test-Mode off
"1" = Test-Mode on
"2" = Fatal Error Messages only

Outputs

RET-CODE str (5)

RET-CODE Values

NTM-TEST-MODE-ON
The test mode value for subsequent messages has been set to "on."

NTM-TEST-MODE-OFF
The test mode value for subsequent messages has been set to "off."

NTM-TEST-MODE-FATAL
The test mode value for subsequent messages has been set for fatal messages only.

NTM-FAILURE
The test status value given in the calling argument is not recognized by the service.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

COBOL

```
MOVE "1" TO TEST-STATUS.
CALL "TSTMOD" USING TEST-STATUS,
                    NTM-RET-CODE.
IF NTM-TEST-MODE-ON
    DISPLAY "TEST MODE IS ENABLED".
```

C LANGUAGE

```
char dbcode;
char rcode[5];

dbcode = '1';
tstmod(&dbcode, rcode);
if (memcmp(rcode, NTM_TEST_MODE_ON, size of (rcode)) == 0)
    printf ("Test mode established");
```

5.1.13 WHTHST

Request the name of the Host on which a given AP resides.

Calling Sequence

```
CALL WHTHST    (AP-NAME,
                HOST-NAME,
                RET-CODE)
```

Description

WHTHST will provide the caller with the name of the Host machine that the specified AP currently resides on.

Inputs

AP-NAME	str (10)

(If blank, the NTM will return the name of the host on which the calling AP resides)

Outputs

HOST-NAME	str (8)
RET-CODE	str (5)

RET-CODE Values

NTM-SUCCESS
The host name for the given AP has been found.

NTM-AP-NOT-FOUND
The given AP name was not found in the NTM's tables.

NTM-FAILURE
The message requesting the AP's Host Name could not be sent.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

COBOL

```
MOVE "TSTAPPL" TO AP-NAME.
MOVE SPACES TO HOST-NAME.
CALL "WHTHST" USING AP-NAME,
                    HOST-NAME,
                    NTM-RET-CODE.
IF NTM-SUCCESS
    THEN DISPLAY AP-NAME " IS ON HOST " HOST-NAME ".
ELSE
    DISPLAY "AP-NAME" NOT FOUND IN NTM TABLES.".
```

C LANGUAGE

```
static char ap_name[11] = "TSTAPPL";
static char host_name[9];
char rcode[5];

whthst(ap_name, host_name, rcode);
if (memcmp(rcode, NTM_SUCCESS, size of (rcode)) != 0)
    printf ("Application not fount in NTM tables");
```

5.2 Service Calls Available to Specialized NTM Applications

These are applications requesting NTM services but not started by the NTM, such as device drivers.

5.2.1 CHGROL

Changes the user's role name during a logon session. This service assumes that an authorization check on the new role has been performed prior to making the change.

Calling Sequence

```
CALL CHGROL  (TERMINAL-ID,
              USER-NAME,
              NEW-ROLE-NAME,
              RET-CODE)
```

Description

The new role name is given to the NTM where it replaces the previous role name in the logon table.

Inputs

TERMINAL-ID	str (8)
USER-NAME	str (8)
NEW-ROLE-NAME	str (10)

Outputs

RET-CODE	str (5)

RET-CODE Values

NTM-SUCCESS
The message informing the Monitor AP of the new role name has been sent successfully.

NTM-FAILURE
The message to the Monitor AP was not sent. The new role has not been entered in the Logon Table.

NTM-USER-NOT-LOGGED-ON
The specified user is not logged on to the NTM. The new role has not been entered in the Logon Table.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

COBOL

```
MOVE NEW-ROLE TO ROLE-NAME.
CALL "CHGROL" USING TERMINAL-ID,
                    USER-NAME,
                    ROLE-NAME,
                    NTM-RET-CODE.
IF NOT NTM-SUCCESS
    MOVE "04" TO GOOF-CODE
    PERFORM SERVICE-ERROR
ELSE
    DISPLAY "NEW ROLE ENTERED IN LOGON TABLE".
```

C LANGUAGE

```
static char termid[9] = "TERMINAL";
static char usrnam[9] = "USERNAME";
static char rolenam[11] = "USERROLE01";
char rcode[5];

chgrol(termid,usrnam,rolenam,rcode);
if (memcmp(rcode, NTM_SUCCESS, size of (rcode)) != 0)
    printf ("Role change failed");
else
    printf ("Role change successful");
```

5.2.2 INITEX

Provide initiation services for an AP.

Calling Sequence

```
CALL INITEX    (BUFFER,
                BUFFER-SIZE,
                RET-CODE)
```

Description

INITEX is a routine called by APs that requires special NTM connection service. It requests that the AP interface perform the necessary initialization to allow the special AP to connect to and communicate with the NTM.

Inputs

BUFFER	str (*)
BUFFER-SIZE	int

Outputs

RET-CODE	str (5)

RET-CODE Values

NTM-SUCCESS
The AP has successfully connected to the NTM.

NTM-FAILURE
The AP could not connect to the NTM.

NTM-RESOURCES-NOT-AVAILABLE
The connection was not made to the NTM. As this is a temporary condition, INITEX may be called again.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

COBOL

```
CALL "INITEX" USING BUFFER,
                    BUFFER-SIZE,
                    RET-CODE.
IF NOT NTM-SUCCESS
    MOVE "01" TO GOOF-CODE
    PERFORM SERVICE-ERROR
ELSE
    NEXT SENTENCE.
```

C LANGUAGE

```
static char ntmbuf[128];
static int ntmsiz = 129;
char rcode[5];

initex(ntmbuf, &ntmsiz, rcode);
if (memcmp(rcode, NTM_SUCCESS, size of (rcode)) != 0)
    {
    printf("Unable to connect to NTM");
    trmnat(CONNECT_ERR);
    }
```

5.2.3 LOGOFF

Allows an AP to inform the NTM of a user logoff.

Calling Sequence

```
CALL LOGOFF  (TERMINAL-ID,
              USER-NAME,
              ROLE-NAME,
              RET-CODE)
```

Description

LOGOFF is used by the application to inform the NTM of a user logoff.

Inputs

TERMINAL-ID	str (8)
USER-NAME	str (8)
ROLE-NAME	str (10)

Outputs

RET-CODE	str (5)

RET-CODE Values

NTM-SUCCESS
The Monitor AP has been sent a message informing it that the User has logged off.

NTM-FAILURE
The message to the Monitor AP was not sent.

NTM-USER-NOT-LOGGED-ON
The user is not logged on to the NTM.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

COBOL

```
CALL "LOGOFF" USING TERMINAL-ID,
                    USER-NAME,
                    ROLE-NAME,
                    RET-CODE.
IF NTM-SUCCESS
    NEXT SENTENCE
ELSE
    DISPLAY "WE'RE DONE BUT WE HAVE A BUG IN LOGOFF.".
```

C LANGUAGE

```
static char termid[9] = "TERMINAL";
static char usrnam[9] = "USERNAME";
static char rolenam[11] = "USERROLE01";
char rcode[5];

logoff(termid,usrnam,rolenam,rcode);
if (memcmp(rcode, NTM_SUCCESS, size of (rcode)) != 0)
    printf ("Logoff failed");
else
    printf ("Logoff successful");
```

5.2.4 LOGON

Allows an AP to pass logon information to the NTM.

Calling Sequence

```
CALL LOGON    (TERMINAL-ID,
               USER-NAME,
               ROLE-NAME,
               RET-CODE)
```

Description

LOGON is used by an AP to provide the NTM with the logon "user" information which the NTM needs to build its logon table. The NTM assumes that the AP has already perfor.ned the required authority checks on the user.

Inputs

TERMINAL-ID	str (8)
USER-NAME	str (8)
ROLE-NAME	str (10)

Outputs

RET-CODE	str (5)

RET-CODE Values

NTM-SUCCESS
The Monitor AP has successfully entered the Logon data in the Logon Table.

NTM-FAILURE
The Monitor AP could not write the new entry to the Logon Table.

NTM-USER-ALREADY-LOGGED-ON
The user is already logged on to the NTM.

NTM_NOT_CONNECTED
The application is not connected to the NTM. NTINIT must be called.

Examples

COBOL

```
*OBTAIN THE USER NAME AND ROLE NAME
*
*INFORM THE NTM OF A SUCCESSFUL LOGON
*
CALL "LOGON" USING TERMINAL-ID,
                   USER-NAME,
                   ROLE-NAME,
                   RET-CODE.
IF NTM-FAILURE
    MOVE "02" TO GOOF-CODE
    PERFORM SERVICE-ERROR
ELSE
    NEXT SENTENCE.
```

C LANGUAGE

```
static char termid[9] = "TERMINAL";
static char usrnam[9] = "USERNAME";
static char rolenam[11] = "USERROLE01";
char rcode[5];

logon(termid,usrnam,rolenam,rcode);
if (memcmp(rcode, NTM_SUCCESS, size of (rcode)) != 0)
    printf ("Logon failed");
else
    printf ("Logon successful");
```

5.3 NTM Service Call Parameters

AP-NAME

The parameter AP-NAME contains a 10-character alphanumeric. The general format of AP names is

DPAPPLNAME

where "DP" is the directory prefix for the directory where the AP's executable module resides, "APPLNAME" is the unique AP name on the specified subsystem. The directory prefix "NT" is reserved.

```
COBOL    01 AP-NAME        PIC X(10).
C        char ap_name [10];
```

BINARY-NATIVE-FLAG

The parameter BINARY-NATIVE-FLAG contains a code which specifies the generic type of data contained in the data portion of a message. Binary (B) indicates that the data is in the host machine's internal representation form whereas native (N) indicates that the data is character data represented by the host machine's character code (ASCII, EBCDIC, etc.).

```
COBOL    01 BINARY-NATIVE-FLAG        PIC X.
C        char binary_native_flag;
```

BUFFER

The parameter BUFFER is the name of the AP area that is passed to the AP Interface on the NTINIT and INITEX calls. The AP Interface uses this area to buffer incoming AP messages. It allows the AP to size this space according to its communication requirements.

```
COBOL    01 BUFFER  PIC X(BUFFER_SIZE)
C        char buffer [buffer_size];
```

BUFFER-SIZE

The parameter BUFFER-SIZE is the size (in bytes) of the buffer area, BUFFER, that is used by the AP Interface to hold incoming AP messages. Guidelines for determining this value are presented in the description of BUFFER.

```
COBOL    01 BUFFER-SIZE   PIC 9(5) COMP.
C        int buffer_size;
```

Any value 1 -- 65535 is supported.

DATA

The parameter DATA contains an alphanumeric which represents the data portion of the message currently being processed.

```
COBOL    01 DATA      PIC X(DATA-LENGTH).
C        char data [data_length];
```

DATA-LENGTH

The parameter DATA-LENGTH contains a numeric value which specifies the length of the data portion of the message currently being processed. Maximum size allowed is 65535.

```
COBOL    01 DATA-LENGTH  PIC 9(5) COMP.
C        int data_length;
```

DESTINATION

The parameter DESTINATION contains a 10-character alphanumeric that is used to specify the destination AP for the message currently being processed. See the description of AP-NAME for more detail.

```
COBOL    01 DESTINATION   PIC X(10).
C        char destination [10];
```

ERROR-CODE

The parameter ERROR-CODE contains the value associated with the error that triggers a call to "SIGERR".

```
COBOL    01 ERROR-CODE    PIC X(5).
C        char error_code [5];
```

ERROR-DESCRIPTION

The parameter ERROR-DESCRIPTION contains information about the error triggering a call to "SIGERR". The content of this parameter is defined by the calling AP.

```
COBOL    01 ERROR-DESCRIPTION  PIC X(72).
C        char error_description [72];
```

HOST-NAME

The parameter HOST-NAME contains an 8-character alphanumeric that is used to specify the "physical" name of the host computer system that the caller is on.

```
COBOL    01 HOST-NAME     PIC X(8).
C        char host_name [8];
```

LOGICAL-CHANNEL

The parameter LOGICAL-CHANNEL contains a value which is used by the APs and the NTM to manage communication paths between APs (see Section 3.1.5). A value must be supplied on a SEND call when the AP wants to pair the send with a response. A blank logical channel is not valid.

```
COBOL    01 LOGICAL-CHANNEL   PIC X(3).
C        char logical_channel [3];
```

MESSAGE-TYPE

The parameter MESSAGE-TYPE contains a code which specifies the message type of the message currently being processed. A blank MESSAGE-TYPE is not valid.

```
COBOL    01 MESSAGE-TYPE      PIC X(2).
C        char message_type [2];
```

OPTIONS

The parameter OPTIONS specifies the characteristics of the transfer and the files.

```
COBOL    01 OPTIONS           PIC X(80).
C        char options[80];
```

REQNO

The parameter REQNO contains a unique identifier for a file transfer request.

```
COBOL    01 REQNO             PIC X(3).
C        char reqno[3];
```

RET-CODE

The parameter RET-CODE contains a value which indicates the return status of a specific request.

```
COBOL    01 RET-CODE          PIC X(5).
C        char ret_code [5];
```

ROLE-NAME

The parameter ROLE-NAME contains a 10-character alphanumeric which identifies the role under which a user is logged on.

```
COBOL    01 ROLE-NAME         PIC X(10).
C        char role_name [10];
```

SEVERITY-LEVEL

The parameter contains a code specifying the level of the error that triggered a call to "SIGERR". Acceptable values are F,E,W, I.

```
COBOL    01 SEVERITY-LEVEL    PIC X.
C        char severity_level;
```

SOURCE

The parameter SOURCE contains a 10-alphanumeric that is used to specify the source AP for a call to RCV with a source match. See the description of AP-NAME for more detail.

```
COBOL    01 SOURCE        PIC X(10).
C        char source [10];
```

SYSTEM-STATE

The parameter SYSTEM-STATE contains a code which indicates the system state of the NTM at the time "NTINIT" is called.

```
COBOL    01 SYSTEM-STATE      PIC X.
C        char system_state;
```

TERMINAL-ID

The parameter TERMINAL-ID identifies the terminal where a given user is logged on.

```
COBOL    01 TERMINAL-ID   PIC X(8).
C        char terminal_id [8];
```

TERMINATION-STATUS

The parameter TERMINATION-STATUS contains a code which indicates the specific condition under which an AP is terminating.

```
COBOL    01 TERMINATION-STATUS      PIC X.
C        char termination_status;
```

TEST-STATUS

The parameter TEST-STATUS contains a code which indicates whether the AP wishes to receive any asynchronous error message, fatal errors only, or not at all. Its description in COBOL is

```
COBOL    01 TEST-STATUS   PIC X.
C        char test_status;
```

TIMEOUT-VALUE

The parameter TIMEOUT-VALUE contains a numeric value which represents a time value in seconds which is used in conjunction with a paired message request. Valid values for timeout are 0 through 86400.

```
COBOL    01 TIMEOUT-VALUE      PIC 9(5) COMP.
C        int timeout_value;
```

USER-NAME

The parameter USER-NAME contains an 8-character alphanumeric value which identifies a specific user.

```
COBOL    01 USER-NAME     PIC X(8).
C        char user_name [8];
```

WAIT-FLAG

The parameter WAIT-FLAG contains a code which indicates whether or not the message currently being processed should have a wait associated with it.

```
COBOL    01 WAIT-FLAG     PIC X.
C        char wait_flag;
```

A value of "0" indicates no waiting in requested and a value of "1" indicates the API should wait.

5.4 Return Code Values

NAME

srvret.h -- NTM application SeRVices RETrun codes

Description

Return codes for verion 1.1 interface.

```
#define NTM_SUCCESS                          "00000"

#define NTM_NO_MESSAGE                       "S0001"
#define NTM_REPLY_REQUIRED_MESSAGE           "S0002"
#define NTM_TIME_OUT_MESSAGE                 "S0003"
#define NTM_TEST_MODE_MESSAGE                "S0004"
#define NTM_HOST_UP                          "S0010"
#define NTM_HOST_DOWN                        "S0011"
#define NTM_TEST_MODE_ON                     "S0020"
#define NTM_TEST_MODE_OFF                    "S0021"
#define NTM_TEST_MODE_FATAL                  "S0022"

#define NTM_FAILURE                          "E0001"
#define NTM_BUFFER_TOO_SMALL                 "E0002"
#define NTM_INVALID_DATA_LENGTH              "E0003"
#define NTM_INVALID_DATA_TYPE                "E0004"
```

```
#define NTM_INVALID_DESGINATION          "E0005"
#define NTM_INVALID_LOGICAL CHANNEL      "E0006"
#define NTM_INVALID_MESSAGE_TYPE         "E0007"
#define NTM_INVALID_SOURCE               "E0008"
#define NTM_INVALID_TIME                 "E0009"
#defin( .NTM_INVALID_TIMEOUT_REQUEST     "E0010"
#define NTM_INVALID_TRG_COND             "E0011"
#define NTM_NOT_AUTHORIZED               "E0012"
#define NTM_AP_NOT_FOUND                 "E0013"
#define NTM_HOST_NOT_KNOWN               "E0014"
#define NTM_NOT_CONNECTED                "E0015"
#define NTM_ALREADY_CONNECTED            "E0016"
#define NTM_USER_NOT_LOGGED_ON           "E0017"
#define NTM_USER_ALREADY_LOGGED_ON       "E0018"
#define NTM_FILE_SOURCE_NAME_INVALID     "E0100"
#define NTM_FILE_DEST_NAME_INVALID       "E0101"
#define NTM_FILE_SOURCE_HOST_INVALID     "E0102"
#define NTM_FILE_DEST_HOST_INVALID       "E0103"
#define NTM_FILE_OPTIONS_INVALID         "E0104"
#define NTM_RESOURCES_NOT_AVAILABLE      "E0900"
#define NTM_RESOURCE_UNAVAIL_HOST        "E0901"
#define NTM_RESOURCE_UNAVAIL_APC         "E0902"
#define NTM_RESOURCE_UNAVAIL_LINK        "E0903"
#define NTM_RESOURCE_UNAVAIL_AP          "E0904"
#define NTM_RESOURCE_UNAVAIL_SYS         "E0905"
#define NTM_RESOURCE_UNAVAIL_MEMORY      "E0906"
```

```
#define NTM_INIT_RECOVERY          "1"
#define NTM_INIT_START             "2"
#define NTM_INIT_NORMAL            "3"
#define NTM_INIT_FIRST_RUN         "4"

#define NTM_TERM_NORMAL            "1"
#define NTM_TERM_SHUTDOWN          "2"
#define NTM_TERM_ABORT             "3"
#define NTM_TERM_EXCEPTION         "4"
```

SRVRET.INC -- FOR VERSION 1.1 OF NTM

```
01 NTM-RET-CODE PIC X(5).

88 NTM-SUCCESS                          "00000".
88 NTM-NO-MESSAGE                       "S0001".
88 NTM-REPLY-REQUIRED MESSAGE           "S0002".
88 NTM-TIME-OUT-MESSAGE                 "S0003".
88 NTM-TEST-MODE-MESSAGE                "S0004".
88 NTM-HOST-UP                          "S0010".
88 NTM-HOST-DOWN                        "S0011".
88 NTM-TEST-MODE-ON                     "S0020".
88 NTM-TEST-MODE-OFF                    "S0021".
88 NTM-TEST-MODE-FATAL                  "S0022".

88 NTM-FAILURE                          "E0001".
88 NTM-BUFFER-TOO-SMALL                 "E0002".
88 NTM-INVALID-DATA-LENGTH              "E0003".
88 NTM-INVALID-DATA-TYPE                "E0004".
88 NTM-INVALID-DESTINATION              "E0005".
88 NTM-INVALID-LOGICAL-CHANNEL          "E0006".
88 NTM-INVALID-MESSAGE-TYPE             "E0007".
88 NTM-INVALID-SOURCE                   "E0008".
88 NTM-INVALIID-TIME                    "E0009".
88 NTM-INVALID-TIMEOUT-REQUEST          "E0010".
88 NTM-INVALID-TRG-COND                 "E0011".
88 NTM-NOT-AUTHORIZED                   "E0012".
88 NTM-AP-NOT-FOUND                     "E0013".
88 NTM-HOST-NOT-KNOWN                   "E0014".
88 NTM-NOT-CONNECTED                    "E0015".
88 NTM-ALREADY-CONNECTED                "E0016".
88 NTM-USER-NOT-LOGGED-ON               "E0017".
88 NTM-USER-ALREADY-LOGGED-ON           "E0018".
88 NTM-FILE-SOURCE-NAME-INVALID         "E0100"
88 NTM-FILE-DEST-NAME-INVALID           "E0101".
88 NTM-FILE-SOURCE-HOST-INVALID         "E0102".
88 NTM-FILE-DEST-HOST-INVALID           "E0103".
88 NTM-FILE-OPTIONS-INVALID             "E0104".
88 NTM-RESOURCES-NOT-AVAILABLE          "E0900".
88 NTM-RESOURCE-UNAVAIL-HOST            "E0901".
88 NTM-RESOURCE-UNAVAIL-APC             "E0902".
88 NTM-RESOURCE-UNAVAIL-LINK            "E0903".
88 NTM-RESOURCE-UNAVAIL-AP              "E0904".
88 NTM-RESOURCE-UNAVAIL-SYS             "E0905"
88 NTM-RESOURCE-UNAVAIL-MEMORY         "E0906".

01 NTM-STATE PIC X.
    88 NTM-INIT-RECOVERY                "1".
    88 NTM-INIT-START                   "2".
    88 NTM-INIT-NORMAL                  "3".
    88 NTM-INIT-FIRST-RUN               "4".
```

SECTION 6

PROGRAMMING GUIDELINES FOR QUEUE SERVER APPLICATIONS

Within the NTM, support is available for specialized applications known as queue servers. The queue server is categorized into two groups:

- o A Type I queue server is an application which is considered an "end-node" in an application chain. This can be seen in Figure 6-1. Application QS1 is a Type I queue server which performs processing for multiple parent applications, (AP1, AP2, ...).

- o A Type II queue server is an application which performs processing for multiple parent applications but also performs the initiation of child applications on behalf of the parent applications. This is illustrated in Figure 6-2.

To support queue server applications, the appplication services have been changed with respect to the use of the LOGICAL-CHANNEL (L.C.).

When a queue server uses the NTRECV service to receive a message, the LOGICAL-CHANNEL parameter returned from the service is a unique "connection identifier". This connection identifier is used by the queue server to identify the application from which the message is received. This unique identifier is required to support the receipt of messages from identically named applications. The identifier is also in all of the "send" services to identify which application the message is to be sent.

The queue server uses the NTQSND service to terminate the connection with the parent application. It may use the NTNSND service to carry on a conversation with a parent but must use the NTQSND service to terminate the connection and allow other parent applications to connect with the queue server. An example of the protocol between a parent application and a queue server is depicted in Figure 6-3. In this example, the parent is sending messages to the queue server over Logical Channel 020. The queue server receives a connection identifier of 001. When the queue server returns a message to the parent via the NTNSND service, the Logical Channel of 020 is returned in the message to the parent. When the queue server returns a message via the NTQSND service, the NTM's knowledge of the connection between the parent and queue server will be removed.

If the parent application (PR1) were to send another message to the queue server after the queue server has replied via the NTQSND service, the message received by the queue server will be identified as a new connection with a new connection identifier.

The NTM maintains a list of the connection identifiers that are being used by the queue server. If an NTM service is used and the LOGICAL_CHANNEL parameter does not identify a valid connection identifier that has been presented to the queue server, the NTM will return a status of NTM_INVALID_LOGICAL_CHANNEL.

As in any application that uses the NTM services, messages are obtained by using the NTRECV service. The receive service support of a queue server is slightly different than that of a normal AP. If the application wishes to receive a message from any source, it uses MESSAGE_SOURCE and LOGICAL_CHANNEL parameters of blanks. When the MESSAGE_SOURCE or LOGICAL_CHANNEL parameters are specified, the queue server NTM support will validate these parameters against the connection identifiers that it maintains and provide selective receipt of messages based on those parameters. If there is no match with any

valid connection identifiers, the NTRECV service will return a status code of NTM_INVALID_LOGICAL_CHANNEL.

To respond to a message that the queue server has received from a parent application, the NTNSND service is used. The LOGICAL_CHANNEL parameter used on this service must be one that has been returned by the NTRECV service. It is by this means that the message is routed to the parent application. The LOGICAL_CHANNEL must match a connection identifier previously returned via the NTRECV service and the parameter MESSAGE_DESTINATION must match a received MESSAGE_SOURCE or a status of NTM_INVALID_LOGICAL_CHANNEL will be returned.

If the message to the parent application is the last that the queue server application will be sending to that particular parent, then the queue server should use the NTQSND service. This service is identical to the NTNSND service except that it terminates the conversation with the parent and removes the connection identifier knowledge from the NTM. The NTQSND service is used only when the destination is a parent application.

The previous paragraphs are applicable to Type I and Type II queue servers. The feature which distinguishes the Type II queue server from the Type I is the user of the NTISND service. A Type II queue server will use the NTISND service to initiate child applications that will do work on behalf of the queue server parent applications.

The NTISND service is used by the queue server to send messages to and cause the initiation of child applications. The NTISND service required a LOGICAL_CHANNEL parameter as input that identifies a connection with a parent application. The NTISND server will return a new connection identifier to the queue server application that will identify the connection with the child application. This connection identifier must be saved by the queue server application to allow further communications with the child application. It is the queue server's responsibibility to correlate the proper parent's connection identifier with the proper child's connection identifier for which the child is performing work. As in normal NTISND processing, the queue server application will be identified as the original source of messages to child applications.

The NTNSND service can now be used in the Type II queue server to send messages to both parent and child applications by specifying the proper connection identifier as input to the LOGICAL_CHANNEL. As in the Type I queue server, the connection identifier will be validated with those known to the NTM.

The queue server application's NTM knowledge of parent connection is removed when the queue server issues a NTQSND request. Its knowledge of child connections is removed when the queue server application receives the application end 'AE' message from the NTM.

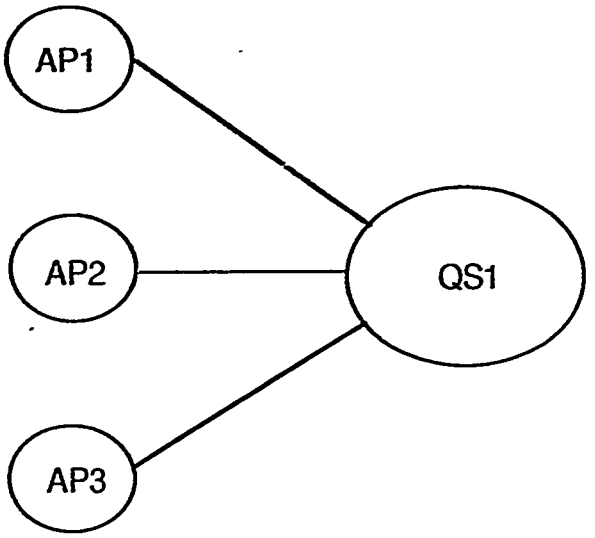


Figure 6-1 QS1: Type I Queue Server for Multiple Parent Applications

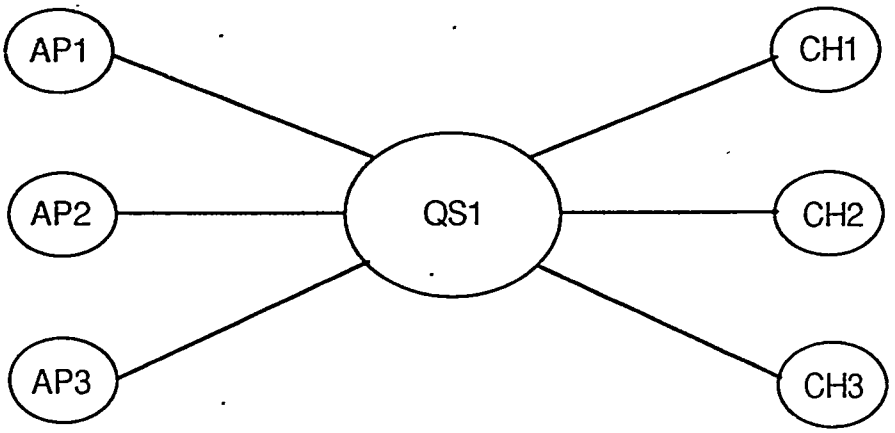


Figure 6-2 QS2: Type II Queue Server for Multiple Parent/Child Applications

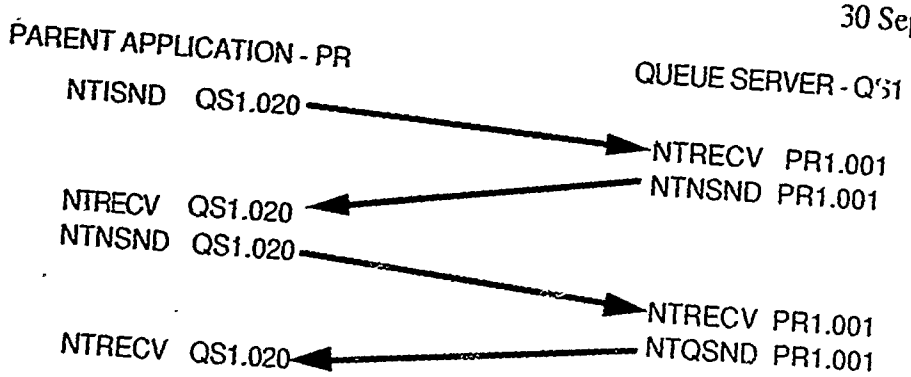


Figure 6-3 Parent Application/Queue Server Protocol

APPENDIX A

MESSAGE FORMATS

Given its characteristics, an AP may receive messages from the NTM. These messages will provide either status information on child APs or system commands. The type of message that is received is indicated by the MESSAGE _TYPE parameter on the NTRECV call. Applications that expect to receive messages from the NTM should include code that processes each message type that could be received.

Each message is described below as to purpose, type, data, and action to be taken by the receiving AP.

Message Type: AP Ending Type ID: AE

Message Purpose:

This message is sent to APs requiring messages on child events. The message informs the parent AP that a child AP has terminated processing.

Data Carried in Data Portion:

Child-AP-Process-Name

AP-Name	PIC X(10)
Instance	PIC X(2)
Child-AP-Channel-ID	PIC X(3)
Child-AP-Status (at Termination)	PIC X

Action by Receiving AP:

The AP may continue processing. If a new instance of the child AP is needed, it may be started using call "NTISND".

Message Type: Cancel Shutdown Type ID: CS

Message Purpose:

Overrides a shutdown pending message.

Data Carried in Data Portion:

None.

Action by Receiving AP:

Display a message to the user that shutdown has been cancelled. The session can then continue.

Message Type: Shutdown AP Type ID: SD & AB

Message Purpose:

Command to the AP to commence its internal shutdown procedures or about procedures.

Data Carried in Data Portion:

None.

Action by Receiving AP:

Begin shutdown procedures. Inform the NTM when shutdown is complete via call "NTTERM" using the NTM_TERM_SHUTDOWN status value.

Message Type: Shutdown Pending Type ID: SP

Message Purpose:

Notification that the NTM will be shutting down in X minutes. This message is sent once a minute until shutdown processing begins.

Data Carried in Data Portion:

Shutdown-Data	PIC X(2) Value "SP"
Time-Until-Shutdown (in Minutes)	PIC X(2)

Action by Receiving AP:

Inform the AP that shutdown will take place in X (Time-Until-Shutdown) minutes. Continue to check for either the next shutdown pending message or a cancel shutdown message.

Message Type: Unsuccessful Initiation **Type ID: NI**

Message Purpose:

Sent when a child AP fails initiation.

Data Carried in Data Portion:

Child-AP-Name	PIC X(10)
Parent-AP-Process-Name	PIC X(12)

Action by Receiving AP:

Terminate any processing requiring the child AP.

Message Type: Timeout Expired **Type ID: TE**

Message Purpose:

Notify application that a response to a paired message has not arrived within the requested timeout interval.

Data Carried in Data Portion:

LOGICAL-CHANNEL	PIC X (3)
ORIGINAL DEST	PIC X (10)

Action by Receiving AP:

Defined by internal AP protocol.

Message Type: File Transfer Status Type ID: XS

Message Purpose:

Notify application that an asynchronous file transfer request has completed.

Data Carried in Data Portion:

RET-CODE	PIC X(5);
REQNO	PIC X(3);
TOFILE	PIC X (80);

Action by Receiving AP:

Defined by internal AP protocol.

APPENDIX B

AP CHARACTERISTICS

In order to integrate new APs into the NTM, the AP Programmer will have to define the AP's message processing features to the NTM AP characteristic record that the NTM will use to determine the correct message handling procedures for the AP. The AP characteristics that have been defined for this AP record for the NTM are the following.

A. AP Participates in NTM Shutdown (Yes/No). The AP programmer must indicate whether the AP performs special cleanup or processing on NTM shutdown. APs that do participate in shutdown must regularly check for "shutdown" messages from the NTM. (Using "NTM" as the source).

B. AP Participates in NTM Recovery (Yes/No). APs that participate in NTM recovery provide recovery logic when they receive a "recover" indicator in the SYSTEM-STATE return of the "NTINIT" call.

C. AP's I/O Characteristics. The AP programmer must characterize the AP's I/O characteristics as one of the following.

1. Does not send to or receive any messages from other NTM APs. (i.e., supports no mailboxes).

2. Sends and receives messages; but sends no messages that require responses. (These APs will receive an error condition if they issue any NTRECV with a non-zero time-out value.)

3. Sends and Receives messages - that require responses as well as ones that may not. This indicates to the NTM that this AP may use NTM message pairing support.

4. Is a "Queue Server" - Can receive messages from multiple APs.

D. AP Handling on Message Time-Outs. The services initially identified are:

1. Receives time-out messages and decides whether to resubmit or terminate. This allows the program to decide whether it wants to wait a longer period of time for a response.

2. Is terminated by the NTM on a time-out.

E. AP Handling on "Child" or "Spawned Task" Termination

1. On normal termination Wants a termination status message (Yes/No)

2. On abnormal termination or abort Wants a termination status message (Yes/No)

3. Require an abort on "child" termination (Yes/No)

The NTM record will also contain information that will be established by the NTM System Administrator with the assistance of the AP Programmer. This includes the.

A. Maximum number of concurrent instances of the AP allowed

B. Maximum number of queued messages allowed for the AP.

APPENDIX C

HELPFUL HINTS

1. Paired Messages must have a non-zero timeout value on the NTNSND or NTISND call. The receiving AP must expect a return of "NTM_REPLY_REQUIRED_ MESSAGE" on its "NTRECV" call.

2. AP's having the characteristic of requiring a specific initiation message must be started with a call to "NTISND". An initiation message sent with a call to "NTNSND" will be rejected for these AP's.

3. A call to "NTMCHK" checks for messages once. If the AP needs to poll (for NTM messages, for example) it must invoke the call periodically. A timer or a counter may be used. As a guideline, when NTM shutdown is pending, a message to that effect is sent every minute until shutdown procedures actually begin.

4. A call to "NTTERM" is the last executable statement in a program. This service ends the AP's execution. The AP should not stop its own run.

5. The logical channel is a critical variable in pairing messages. It must be specified whenever the sending AP wants to ensure that it will receive the correct response from the correct instance of the responding AP. See Section 3.1.5 for details.

6. On any "send message" call, the sending AP must specify the destination, message type, and data length. There are no default values for these data items.

7. When a call to "NTMCHK" returns the information that a message has been found, the AP can immediately call "NTRECV" to retrieve the message.

8. AP's invoking any service must, in some way, handle all possible return codes.

9. When the AP wants to receive a message from the NTM, it must specify "NTM" as the source in the calling parameters. This holds for both "NTRECV" and "NTMCHK".

10. Message Types may be defined by the AP programmer.

11. If an AP wishes to receive asynchronous error messages, a call to "NTTSMD" is required to set test mode to on.

APPENDIX D

API LINKING INSTRUCTIONS

Section 8 of the NTM Installation Guide describes the link templates available for applications calling the NTM services.

APPENDIX E

NTM SAMPLE PROGRAMS

RECEIVE PROGRAM IN C:

```
/* NAME
 *   tsmsgrt - TeSt MeSsaGe check ReTurn program
 */

#include "stdtyp.h"
#include "stdio.h"
#include "string.h"
#include "stdlib.h"
#include "ntmapi.h"


main()
  {
  char sysstate;
  int  ret;
  char src[10];
  char schan[3];
  char stype[2];
  char buf[4096];
  int  buflen;
  static const int bufsiz = sizeof buf;

  if ((ret = ntinit(&sysstate)) != NTM_SUCCESS)
     {
     printf("ntinit failed - %d\n", ret);
     ntterm(NTM_TERM_INIT_FAIL);
     }
  for (;;)
     {
     memset(src, ' ', sizeof src);
     memset(schan, ' ', sizeof schan);
     memset(stype, ' ', sizeof stype);
     if ((ret = ntrecv(src, schan, stype, "1", bufsiz, &buflen, buf)) !=
          NTM_SUCCESS)
        {
        printf("ntrecv failed - %d\n", ret);
        break;
        }
     if (memcmp(src, "NTM       ", sizeof src) == 0) break;
     if ((ret = ntnsnd(src, schan, stype, 'N', 0, buflen, buf)) != NTM_SUCCESS)
        {
        printf("ntnsnd failed - %d\n", ret);
        break;
        }
     }
  ntterm(NTM_TERM_NORMAL);
  }
```

SEND PROGRAM IN C:

```
/* NAME
 *    tsmsgck - TeSt MeSsaGe CheckK program
 */

#include "stdtyp.h"
#include "stdio.h"
#include "string.h"
#include "stdlib.h"
#include "signal.h"
#include "ntmapi.h"

static bool done = FALSE;
static void handler PROTO((int));


main()
  {
  char sysstate;
  char rcode[5];
  int  ret;
  char orgsrc[15];
  char uiname[10];
  char uichan[3];
  char uiterm[8];
  char uiuser[8];
  char uirole[10];
  char host[8];
  char start[23];
  int  osstat;
  char newtyp[2];
  int  loop, good, bad;
  char src[10];
  char schan[3];
  char stype[2];
  int  buflen;
  const char *p, *q;
  static char buf[4096];
  static const char term[] = "THISTERM";
  static const char user[] = "NTMSGCHK";
  static const char role[] = "TESTDRIVER";
  static const char chartab[] =
    "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
  static const char dest[] = "NTTSMSGRET";
  static const char dchan[] = "001";
  static const char dbuff[] = "TESTING ANOTHER MESSAGE";
  static const int dblen = sizeof dbuff - 1;
  static const int bufsiz = sizeof buf;

  if ((ret = ntxcon(&sysstate)) != NTM_SUCCESS)
    {
    printf("ntxcon failed - %d\n", ret);
    ntterm(NTM_TERM_INIT_FAIL);
```

```
  }
ret = ntgtos(orgsrc);
printf("OS NAME-> %.15s RC-> %d\n", orgsrc, ret);
if ((ret = ntulon(term, user, role)) != NTM_SUCCESS)
  {
  printf("ntulon failed - %d\n", ret);
  ntterm(NTM_TERM_EXCEPTION);
  }
ret = ntuget(uiname, uichan, uiterm, uiuser, uirole);
printf("AP Name = %.10s, Chan = %.3s, ", uiname, uichan);
printf("Term = %.8s, User = %.8s, Role = %.10s, ", uiterm, uiuser, uirole);
printf("Ret = %d\n", ret);
ret = ntwhst(dest, host);
printf("Echo will run on host %.8s (Ret = %d)\n", host, ret);
asctim(start, rcode, &osstat);
if (memcmp(rcode, "00000", sizeof rcode) == 0)
  printf("Test starting at %.23s\n", start);
else
  printf("ASCTIM failed - %d, %08x\n", rcode, osstat);
loop = good = bad = 0;
sysstate = NTM_TERM_EXCEPTION;
signal(SIGINT, handler);
for (p = chartab; *p && !done; p++)
  {
  newtyp[0] = *p;
  for (q = chartab; *q && !done; q++)
    {
    newtyp[1] = *q;
    if ((ret = ntnsnd(dest, dchan, newtyp, 'N', 0, dblen, dbuff)) !=
        NTM_SUCCESS)
      {
      printf("ntnsnd failed - %d\n", ret);
      goto err;
      }
    memset(src, ' ', sizeof src);
    memset(schan, ' ', sizeof schan);
    memset(stype, ' ', sizeof stype);
    if ((ret = ntrecv(src, schan, stype, "1", bufsiz, &buflen, buf)) !=
        NTM_SUCCESS)
      {
      printf("ntrecv failed - %d\n", ret);
      goto err;
      }
    if (memcmp(src, "NTM      ", sizeof src) == 0)
      {
      printf("Received %2.2s message from NTM. Shutting Down\n", stype);
      done = TRUE;
      }
    if (memcmp(stype, newtyp, sizeof stype) == 0) ++good;
    else
      {
      printf("Expected -> %.2s got %.2s\n", newtyp, stype);
      ++bad;
      }
```

```
        }
      asctim(start, rcode, &osstat);
      printf("Loop %d finish at %.23s\n", ++loop, start);
      }
    if ((ret = ntsabt(dest, dchan)) != NTM_SUCCESS)
      printf("ntsabt failed - %d\n", ret);
    sysstate = NTM_TERM_NORMAL;
err:
    if ((ret = ntulof()) != NTM_SUCCESS)
      printf("ntulof failed - %d\n", ret);
    printf("Message type checking is complete.\n\n");
    printf("Number of good messages -> %d\n", good);
    printf("Number of bad messages  -> %d\n", bad);
    ntterm(sysstate);
    }
```

```
static void handler(sig)
  int sig;
  {

  done = TRUE;
  }
```


☆U.S. Government Printing Office:1992—648-127/62243